Predicción del valor de las viviendas en California mediante algoritmos de Machine Learning

Juan Menduiña[1]

Resumen

El presente trabajo analiza el desempeño de diversos algoritmos de *Machine Learning* en la predicción del valor de las viviendas en California. Se evalúa la capacidad predictiva de varios modelos de aprendizaje supervisado, entre ellos *K-Nearest Neighbors (kNN)*, *Ridge Regression*, *Lasso Regression*, *Elastic Net*, *Random Forest* y *Gradient Boosting Trees*, además de un enfoque de *Stacking* mediante el algoritmo *Super Learner*. El principal hallazgo del trabajo es que el enfoque de *Stacking*, que aprende cómo combinar las predicciones de todos estos modelos para obtener una predicción más precisa, es el que mejor predice el valor de las viviendas, considerando diferentes métricas para evaluar la calidad de las predicciones.

Palabras clave: aprendizaje automático, aprendizaje supervisado, stacking.

Abstract

This paper analyzes the performance of various Machine Learning algorithms in predicting home values in California. The predictive ability of several supervised learning models is evaluated, including K-Nearest Neighbors (kNN), Ridge Regression, Lasso Regression, Elastic Net, Random Forest, and Gradient Boosting Trees, as well as a Stacking approach using the Super Learner algorithm. The main finding of the paper is that Stacking approach, which learns how to combine the predictions of all these models to obtain a more accurate prediction, is the one that best predicts the value of homes, considering different metrics to evaluate the quality of the predictions.

Keywords: machine learning, supervised learning, stacking.

[1] Licenciado y Magíster en Economía - UNLP. E-mail address: menduinajuan@gmail.com.

1. Introducción

El uso de algoritmos de aprendizaje supervisado ha revolucionado la predicción en diversos campos (Hastie *et al.*, 2009; James *et al.*, 2013). En particular, la predicción del valor de las viviendas resulta clave para la toma de decisiones en sectores como el inmobiliario, financiero y gubernamental.

En este trabajo, se analiza el desempeño de distintos algoritmos de *Machine Learning* en la predicción del valor de las viviendas en California (EE.UU.), utilizando un conjunto de datos que contiene información sobre diversas características socioeconómicas y geográficas de diferentes zonas en 1990. En particular, se evalúa la capacidad predictiva de varios modelos de aprendizaje supervisado, entre ellos *K-Nearest Neighbors (kNN)*, *Ridge Regression*, *Lasso Regression*, *Elastic Net*, *Random Forest* y *Gradient Boosting Trees*, además de un enfoque de *Stacking* mediante el algoritmo *Super Learner*.

El principal hallazgo del trabajo es que el enfoque de *Stacking*, que aprende cómo combinar las predicciones de todos estos modelos para obtener una predicción más precisa, es el que mejor predice el valor de las viviendas en California, considerando diferentes métricas para evaluar la calidad de las predicciones (error absoluto medio, raíz del error cuadrático medio, raíz del error logarítmico cuadrático medio y coeficiente de determinación).

El resto de este trabajo está estructurado de la siguiente manera. En la sección 2, se describen la fuente y el procesamiento de datos. En la sección 3, se analizan las correlaciones entre las variables. En la sección 4, se describen los modelos de aprendizaje supervisado utilizados. En la sección 5, se analizan los resultados obtenidos. Finalmente, en la sección 6, se concluye.

2. Fuente y procesamiento de datos

El *dataset* que se utiliza en el trabajo se obtiene de *scikit-learn*, que contiene el valor medio de las viviendas de diferentes zonas de California en 1990 y, además, ocho variables (*features*) que representan diversas características de éstas. La base dispone, originalmente, de 20.640 observaciones. A continuación, se describen las variables:

- *MedHouseVal:* valor medio de las viviendas de la zona, en cientos de miles de dólares.
- *MedInc:* ingreso medio de los viviendas de la zona, en decenas de miles de dólares.
- *HouseAge:* edad media de las viviendas de la zona, en años.
- *AveRooms:* número promedio de habitaciones por vivienda de la zona.
- *AveBedrms:* número promedio de dormitorios por vivienda de la zona.
- *Population:* cantidad total de personas que viven en la zona.
- *AveOccup:* promedio de personas por vivienda en la zona.
- *Latitude:* latitud de la zona.
- *Longitude:* longitud de la zona.

Cabe aclarar que todas las *features* son numéricas y que no existen valores nulos en ninguno de ellas ni filas duplicadas en el *dataset*.

2.1. *Tratamiento de outliers*

En la Tabla 1, se presenta un resumen estadístico del *dataset*, en donde, a simple vista, se puede observar la existencia de valores extremos. Por ejemplo, el máximo de *AveRooms* y *AveOccup* es 141,91 y 1.243,33, respectivamente, lo cual es demasiado alto. Por lo tanto, con el fin de depurar de *outliers* el *dataset*, se utiliza el método IQR, el cual consiste en eliminar aquellas observaciones que se encuentren a una distancia superior a 1,5 veces su IQR de los cuartiles 1 y 3. Luego de esta depuración, el *dataset* se reduce a 16.312 observaciones.

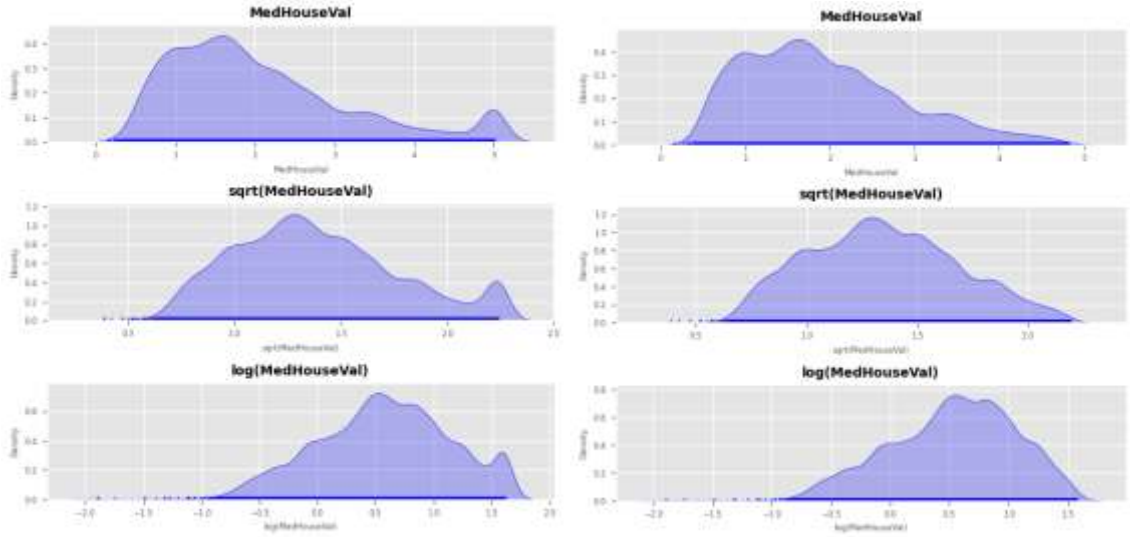
Tabla 1. *Estadísticas descriptivas del dataset.*

Variable	Media	Desvío	Mínimo	25%	50%	75%	Máximo
MedHouseVal	2,07	1,15	0,15	1,20	1,80	2,65	5,00
MedInc	3,87	1,90	0,50	2,56	3,53	4,74	15,00
HouseAge	28,64	12,59	1,00	18,00	29,00	37,00	52,00
AveRooms	5,43	2,47	0,85	4,44	5,23	6,05	141,91
AveBedrms	1,10	0,47	0,33	1,01	1,05	1,10	34,07
Population	1425,48	1132,46	3,00	787,00	1166,00	1725,00	35682,00
AveOccup	3,07	10,39	0,69	2,43	2,82	3,28	1243,33
Latitude	35,63	2,14	32,54	33,93	34,26	37,71	41,95
Longitude	-119,57	2,00	-124,35	-121,80	-118,49	-118,01	-114,31

Fuente: Elaboración propia.

En las Figuras 1 y 2, se puede observar la distribución de la variable *target* y de las *features*, respectivamente, antes y después de la eliminación de *outliers*.

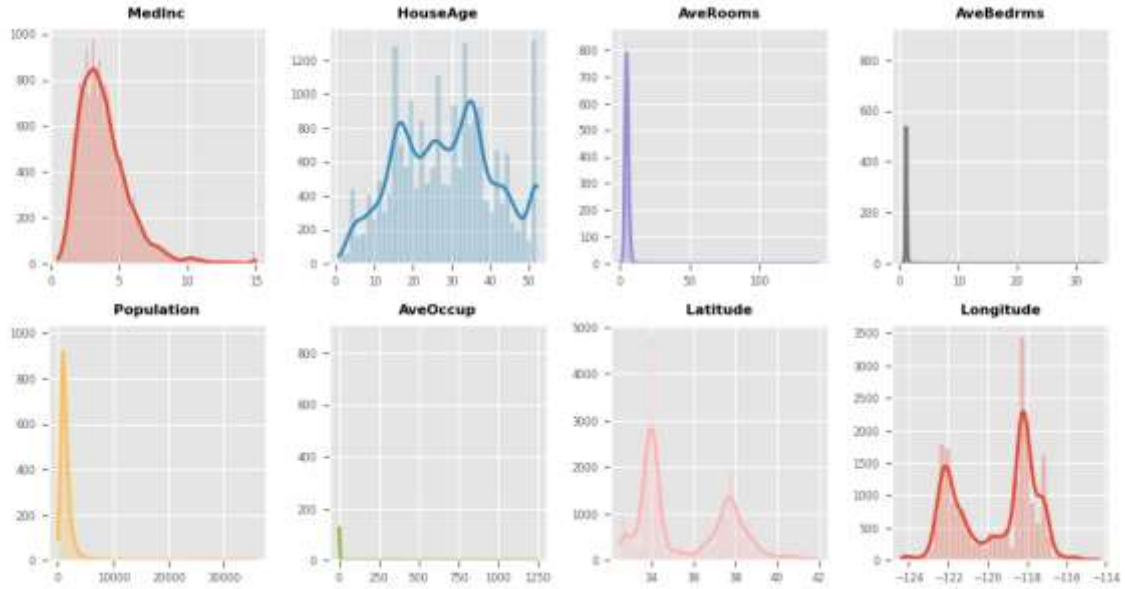
Figura 1. *Distribución de la variable target, antes y después de la eliminación de outliers.*



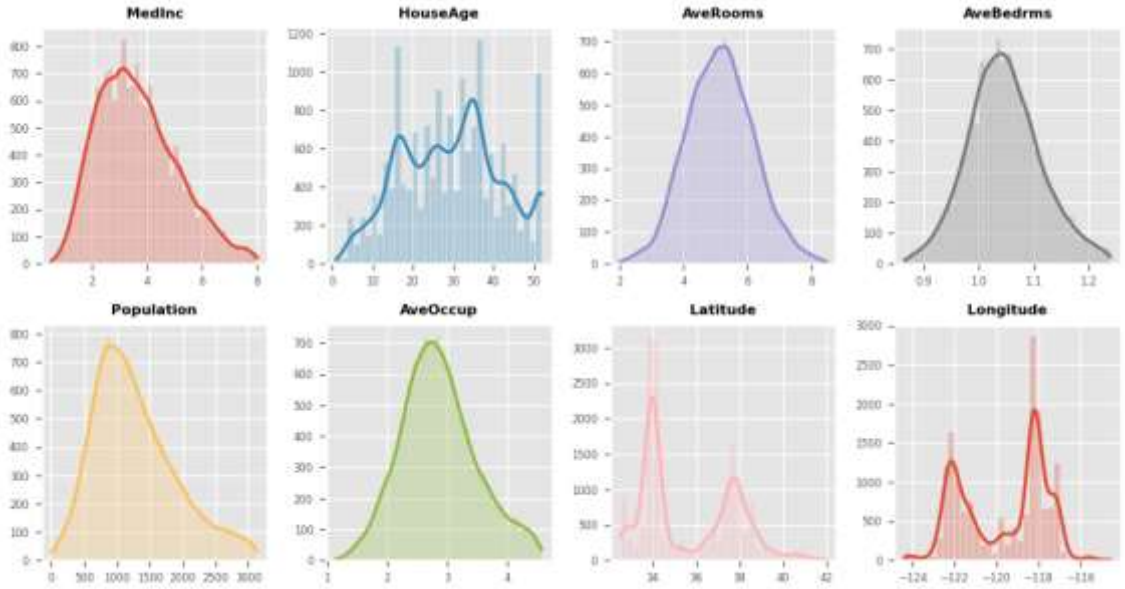
(a) Antes **(b)** Después

Fuente: Elaboración propia.

Figura 2. *Distribución de las features, antes y después de la eliminación de outliers.*



(a) Antes



(b) Después

Fuente: Elaboración propia.

2.2. *Estandarización de las features numéricas*

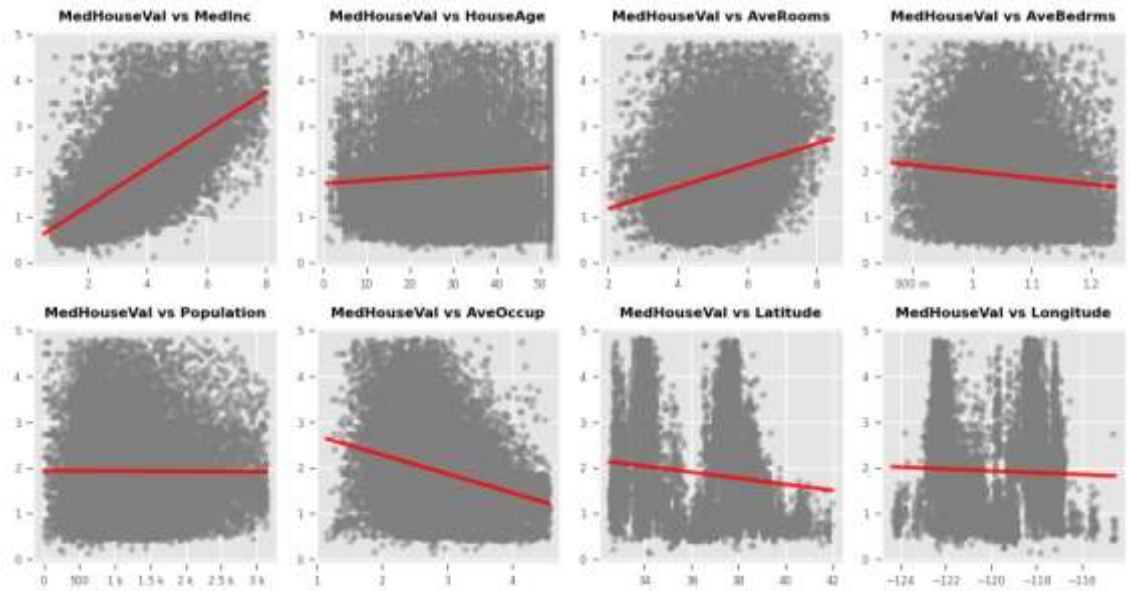
Cuando las *features* son numéricas, la escala en la que se miden así como la magnitud de su varianza pueden influir, en gran medida, en algunos algoritmos de *Machine Learning*, de forma que aquellas *features* que se midan en una escala mayor o que tengan mayor varianza dominarán el modelo aunque no sean las que más relación tienen con la variable

target. Por lo tanto, es necesario estandarizar las *features*. En particular, éstas se normalizan, restándole a cada una su media y dividiendo esto por su desviación estándar.[2]

3. Análisis de la variable *target* y las *features*

En esta sección, se presenta la correlación existente entre las variables. En la Figura 3, se presenta un *scatterplot* de la variable *target* con cada *feature*, en donde se observa que parece existir una relación positiva con *MedInc*, *HouseAge* y *AveRooms* y una relación negativa con *AveBedrms* y *AveOccup*. Esto también se puede apreciar en el *heatmap* de la Figura 4, donde, además, se presentan las correlaciones entre las *features*.

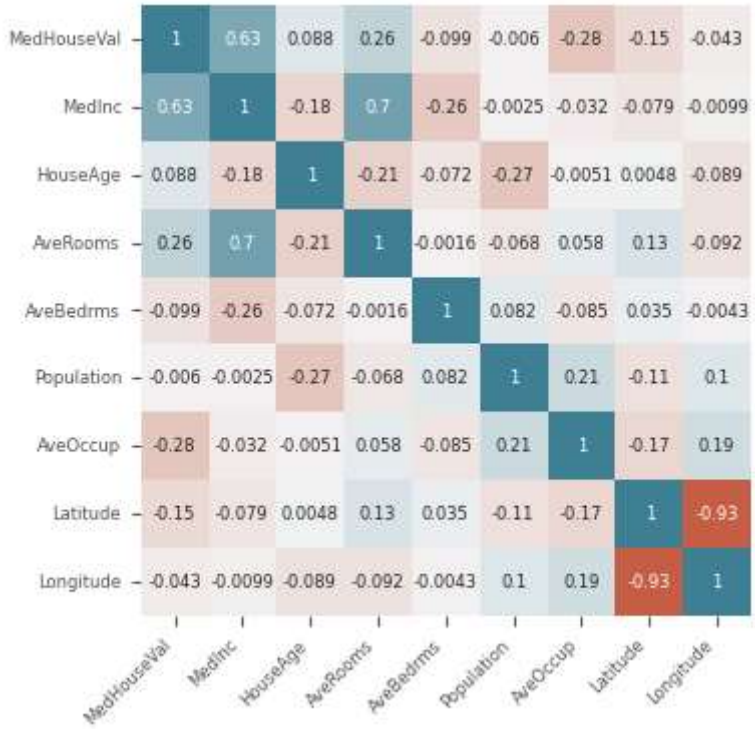
Figura 3. *Correlación entre la variable target y las features.*



Fuente: Elaboración propia.

[2] Cabe aclarar que la estandarización se realiza luego de la división del *dataset* en partición de entrenamiento (*train*) y partición de validación (*test*). Por un lado, a la hora de entrenar un modelo, para cada validación cruzada, en primer lugar, los datos del conjunto de entrenamiento (*folds* de entrenamiento) se estandarizan dentro de cada *fold* usando la media y la desviación estándar de este conjunto y, luego, los datos del *fold* de validación se estandarizan usando las estadísticas del conjunto de entrenamiento. Por otro lado, luego de obtenida la combinación óptima de hiperparámetros y de re-entrenado el modelo con toda la partición de entrenamiento, a la hora de realizar predicciones, la partición de validación del *dataset* se estandariza usando la media y la desviación estándar de la partición de entrenamiento.

Figura 4. *Heatmap de correlaciones.*



Fuente: Elaboración propia.

4. Algoritmos de *Machine Learning*

En esta sección, se describen, en términos muy generales, los algoritmos de *Machine Learning* utilizados en este trabajo para predecir el valor de las viviendas en California. Como ya fue mencionado, se evalúa la capacidad predictiva de los siguientes modelos de aprendizaje supervisado: *K-Nearest Neighbors (kNN)*, *Ridge Regression*, *Lasso Regression*, *Elastic Net*, *Random Forest*, *Gradient Boosting Trees* y *Stacking* (algoritmo *Super Learner*).[3] Cada uno de estos algoritmos aporta diferentes fortalezas y debilidades en términos de sesgo, varianza y capacidad de generalización (Hastie *et al.*, 2009; James *et al.*, 2013), permitiendo evaluar cuál es el más adecuado para la predicción del valor de las viviendas en California.

4.1. *Modelo K-Nearest Neighbors (kNN)*

El modelo *K-Nearest Neighbors (kNN)* es un algoritmo basado en la proximidad de los datos. Para realizar una predicción, el modelo identifica los *k* puntos de entrenamiento

[3] Cabe mencionar que todo el trabajo se realiza en Python, haciendo uso de la biblioteca *scikit-learn*, que es la biblioteca más destacada de *Machine Learning* en Python (Pedregosa *et al.*, 2011).

más cercanos a una observación de prueba y toma un promedio de sus valores objetivo. La elección del parámetro k es crucial, ya que valores pequeños pueden llevar a sobreajuste, mientras que valores grandes pueden generar subajuste. Este modelo es altamente influenciado por la escala de los datos, por lo que es fundamental estandarizar las *features* antes de su aplicación.

4.2. *Modelos Ridge, Lasso y Elastic Net*

Los modelos de regresión *Ridge*, *Lasso* y *Elastic Net* son extensiones de la regresión lineal que incorporan penalizaciones para evitar problemas de sobreajuste.

- *Ridge Regression* agrega una penalización L2 al error cuadrático medio, lo que reduce la magnitud de los coeficientes sin eliminarlos por completo, lo cual es útil cuando las *features* están correlacionadas.
- *Lasso Regression* agrega una penalización L1 al error cuadrático medio, lo que puede forzar algunos coeficientes a ser, exactamente, cero, facilitando la selección de variables.
- *Elastic Net* combina ambas penalizaciones (L1 y L2), proporcionando mayor flexibilidad y mejor desempeño en conjuntos de datos con muchas *features* correlacionadas.

4.3. *Modelo Random Forest*

Random Forest es un modelo basado en ensamblado de múltiples árboles de decisión. Cada árbol se entrena sobre una muestra aleatoria del conjunto de datos y selecciona un subconjunto de *features* en cada división. Las predicciones finales se obtienen promediando las predicciones individuales de todos los árboles. Este método es robusto frente al sobreajuste y maneja bien datos con alta dimensionalidad y relaciones no lineales.

4.4. *Modelo Gradient Boosting Trees*

Gradient Boosting Trees es otro método basado en árboles de decisión, pero, en lugar de entrenar múltiples árboles independientes, los construye secuencialmente. Cada árbol corrige los errores cometidos por los anteriores mediante la minimización del gradiente

del error. Modelos como *XGBoost* y *LightGBM* son versiones optimizadas de esta técnica y se han convertido en estándar en muchas competencias de predicción debido a su alta precisión y capacidad de ajuste.

4.5. *Stacking (algoritmo Super Learner)*

El *Stacking* es un enfoque de ensamblado que combina múltiples modelos base para mejorar la capacidad predictiva. Se basa en entrenar diferentes algoritmos y, luego, utilizar un meta-modelo que aprende a combinar sus predicciones de manera óptima (Wolpert, 1992; Zhou, 2012). En este trabajo, el enfoque de *Stacking*, mediante el algoritmo *Super Learner*, se construye utilizando los modelos previamente mencionados como base *learners* (considerando la combinación óptima de hiperparámetros que se obtuvo del entrenamiento de cada modelo) y un modelo adicional (*Ridge Regression*, en este trabajo)[4] como meta-modelo para combinar las predicciones de estos.

5. Entrenamiento y evaluación de modelos

Los modelos descritos en la sección anterior se entrenan con la partición de entrenamiento del *dataset*, excluyendo de este entrenamiento a la partición de validación, como es habitual en estos algoritmos.[5] Para ello, se consideran 50 combinaciones aleatorias de hiperparámetros, seleccionadas a partir de un conjunto discrecional de H combinaciones posibles (que depende de cada modelo),[6] y se emplea validación cruzada repetida (con 5 *folds* y 4 repeticiones).

Dado que cada combinación considerada de hiperparámetros se evalúa en 20 *folds*, el total de ajustes realizados por modelo asciende a 1.000. A su vez, para cada modelo, la capacidad predictiva de cada combinación de hiperparámetros se mide a través del promedio (sobre los 20 *folds*) de la raíz del error cuadrático medio. Como resultado de este proceso, se obtiene la combinación óptima de hiperparámetros para cada modelo.[7] Luego, con estos hiperparámetros óptimos, cada modelo se entrena nuevamente

[4] Se utiliza el modelo *RidgeCV* del módulo *sklearn.linear_model* dentro de la biblioteca *scikit-learn* de Python, el cual selecciona automáticamente el mejor valor de α usando validación cruzada.
[5] Se selecciona, de manera aleatoria, el 80% del *dataset* como partición de entrenamiento y el 20% como partición de validación.
[6] Ver el código proporcionado de Python para más detalles sobre los valores posibles que se decidió que puede tomar cada hiperparámetro de cada modelo.
[7] En la Tabla A1 del Apéndice, se presentan cuáles resultaron ser estos hiperparámetros óptimos para cada modelo.

utilizando toda la partición de entrenamiento, estimando, así, sus parámetros finales (pesos en regresión, nodos en árboles, etc.).

Para cada modelo propuesto, finalizado el entrenamiento, sobre la base de la combinación óptima de hiperparámetros y la estimación correspondiente de parámetros, se utiliza la partición de validación para realizar predicciones. Por último, con el fin de comparar la capacidad predictiva de los diferentes modelos de aprendizaje supervisado entrenados, se utilizan diferentes métricas sobre la partición de validación: error absoluto medio (MAE), raíz del error cuadrático medio (RMSE), raíz del error logarítmico cuadrático medio (RMSLE) y coeficiente de determinación (R^2).

En la Tabla 2, se presentan las métricas calculadas sobre la partición de validación para cada modelo. En primer lugar, se puede observar que el enfoque de *Stacking* es el que mejor predice el valor de las viviendas en California, considerando cualquiera de las métricas utilizadas para evaluar la calidad de las predicciones. En particular, se tiene que el RMSE en *Stacking* es 0,3926, menor al 0,3969 correspondiente a *Gradient Boosting Trees*, el siguiente modelo con menor RMSE.

Por otro lado, se podría decir que *Gradient Boosting Trees* es el segundo modelo con mayor poder predictivo, ya que es el que obtiene el segundo menor valor de cada métrica (segundo mayor en el caso del R^2). Finalmente, los modelos lineales (*Ridge*, *Lasso* y *Elastic Net*) presentan el menor poder predictivo, lo cual es esperable considerando que la relación entre la variable *target* y las *features* se espera que sea altamente no lineal y estos modelos no capturan patrones no lineales que pueden existir en los datos. En líneas generales, se observa que el ordenamiento de "mejor a peor" modelo, en términos de capacidad predictiva, es robusto a la elección de la métrica.

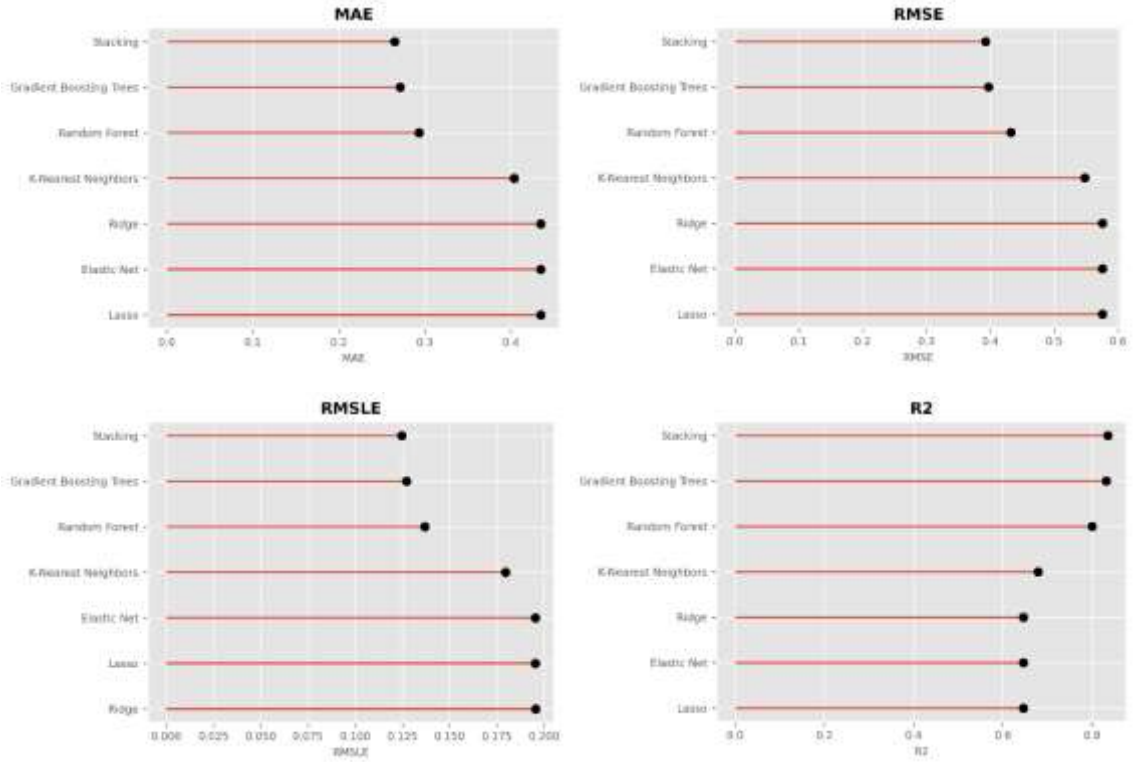
Tabla 2. *Métricas sobre la partición de validación para cada modelo.*

Modelo	**MAE**	**RMSE**	**RMSLE**	R^2
1. *K-Nearest Neighbors*	0,4036	0,5478	0,1796	0,6789
2. *Ridge*	0,4344	0,5750	0,1954	0,6462
3. *Lasso*	0,4345	0,5751	0,1954	0,6461
4. *Elastic Net*	0,4344	0,5750	0,1954	0,6462
5. *Random Forest*	0,2931	0,4320	0,1368	0,8003
6. *Gradient Boosting Trees*	0,2713	0,3969	0,1271	0,8314
7. *Stacking*	0,2649	0,3926	0,1247	0,8351

Fuente: Elaboración propia.

Lo dicho anteriormente también se puede observar más claramente, de manera gráfica, en la Figura 5, en donde se presentan, para cada métrica calculada sobre la partición de validación, el valor que se obtiene para cada modelo.

Figura 5. *Métricas sobre la partición de validación para cada modelo.*



Fuente: Elaboración propia.

6. Conclusiones

En este trabajo, se analizó el desempeño de distintos algoritmos de *Machine Learning* en la predicción del valor de las viviendas en California (EE.UU.), utilizando un conjunto de datos que contiene información sobre diversas características socioeconómicas y geográficas de diferentes zonas en 1990. En particular, se evaluó la capacidad predictiva de varios modelos de aprendizaje supervisado, entre ellos *K-Nearest Neighbors (kNN)*, *Ridge Regression*, *Lasso Regression*, *Elastic Net*, *Random Forest* y *Gradient Boosting Trees*, además de un enfoque de *Stacking* mediante el algoritmo *Super Learner*.

El principal hallazgo del trabajo fue que el enfoque de *Stacking*, que aprendió cómo combinar las predicciones de todos estos modelos para obtener una predicción más precisa, fue el que mejor predijo el valor de las viviendas en California, considerando diferentes métricas para evaluar la calidad de las predicciones (MAE, RMSE, RMSLE,

R^2). Además, cabe mencionar que, de los modelos base, *Gradient Boosting Trees* fue el de mayor poder predictivo, mientras que la peor *performance* la tuvieron los modelos lineales (*Ridge*, *Lasso* y *Elastic Net*).

No obstante, futuras investigaciones podrían evaluar: (i) variaciones en los modelos base que se utilizaron en el *Stacking*, sobre todo en la elección del conjunto de combinaciones posibles de hiperparámetros y en la elección de la cantidad de éstas que se seleccionaron aleatoriamente para ajustar cada modelo; (ii) así como también una elección distinta de meta-modelo para el *Stacking*, pudiendo seleccionar, por ejemplo, algún modelo no lineal. Esto con el fin de explorar posibles cambios en las conclusiones en cuanto a la capacidad predictiva de estos modelos, considerando que los resultados a favor del *Stacking*, eventualmente, están sujetos a las elecciones discrecionales que hubo que realizar durante el trabajo.[8]

[8] Por ejemplo, dada la limitada capacidad de cómputo, tanto por cuestiones de *hardware* como de *software*, en ningún modelo se realizó una búsqueda exhaustiva sobre el conjunto discrecional de H combinaciones posibles de hiperparámetros (sino que se evaluaron 50 combinaciones aleatorias) ni tampoco se incrementó el tamaño de este conjunto discrecional (es decir, no se consideraron más combinaciones posibles). Ambas cosas impidieron ajustar el modelo con una mayor cantidad de combinaciones de hiperparámetros y, potencialmente, encontrar una combinación que ajuste mejor a los datos.

Referencias

Hastie, T., Tibshirani, R. y Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

James, G., Witten, D., Hastie, T., Tibshirani, R. y Taylor, J. (2013). An Introduction to Statistical Learning: With Applications in Python. Springer.

Pedregosa, F. *et al.* (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830. Recuperado de https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

Wolpert, D. H. (1992). Stacked Generalization. *Neural Networks*, 5, 241-259. https://doi.org/10.1016/S0893-6080(05)80023-1

Zhou, Z. H. (2012). Ensemble Methods: Foundations and Algorithms. CRC Press.

Apéndice

Tabla A1. *Hiperparámetros óptimos para cada modelo.*

Hiperparámetro	***K-Nearest Neighbors***	***Ridge***	***Lasso***	***Elastic Net***	***Random Forest***	***Gradient Boosting Trees***
n_neighbors	19					
alpha		0,716839	0,000195	0,000226		
l1_ratio				0,248996		
n_estimators					500	500
max_features					4	5
max_depth					20	5
subsample						1

Fuente: Elaboración propia. Nota: Se utiliza la nomenclatura de la biblioteca *scikit-learn* para los nombres de los hiperparámetros (ver código proporcionado de Python).