UNIVERSIDAD
DE GRANADA

Inteligencia de Negocio

PRÁCTICA 3

Competición en Kaggle

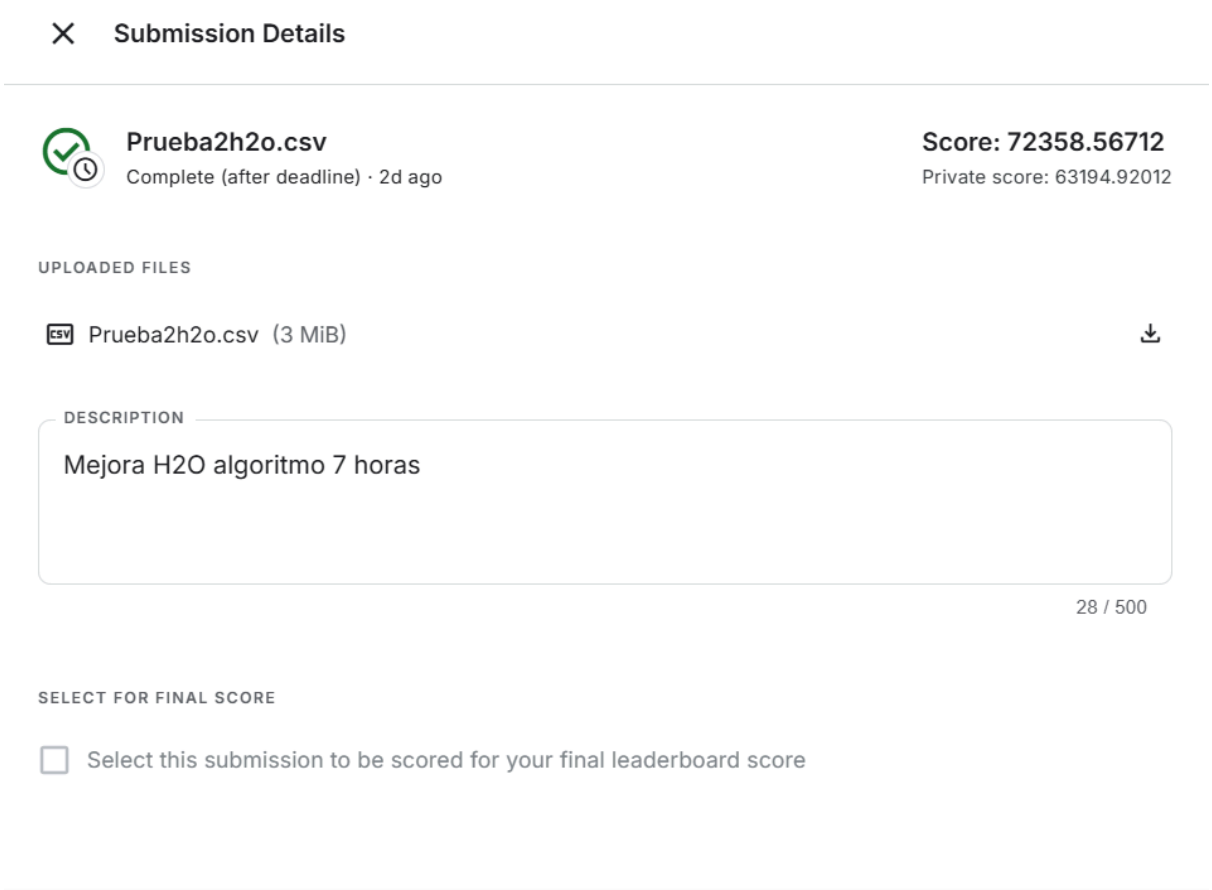
José Antonio Fernández Aranda

Subgrupo 1

jantoniofer@correo.ugr.es

Índice

Puntuación Obtenida en Kaggle



Introducción

En esta práctica de la asignatura de Inteligencia de Negocio, he trabajado en una competición de Kaggle cuyo objetivo es predecir el precio de coches usados utilizando técnicas de aprendizaje supervisado. A través de este trabajo, he aplicado los conocimientos adquiridos en el curso sobre procesamiento de datos, construcción de modelos predictivos y evaluación de resultados.

La tarea ha consistido en analizar un conjunto de datos con características categóricas y numéricas, implementar diferentes algoritmos de regresión y optimizar sus parámetros para obtener los mejores resultados posibles. Además, he aprendido a utilizar Kaggle como plataforma para comparar mis predicciones con otros participantes, lo que me ha permitido mejorar iterativamente mis modelos.

A nivel personal, esta práctica me ha ayudado a entender mejor cómo trabajar con datos reales y a enfrentarme a los problemas que pueden surgir en el camino, como valores faltantes, la selección de variables y el ajuste de los modelos.

Además también se ha realizado la configuración de la plataforma Kaggle, tal y como se comentó en clase:

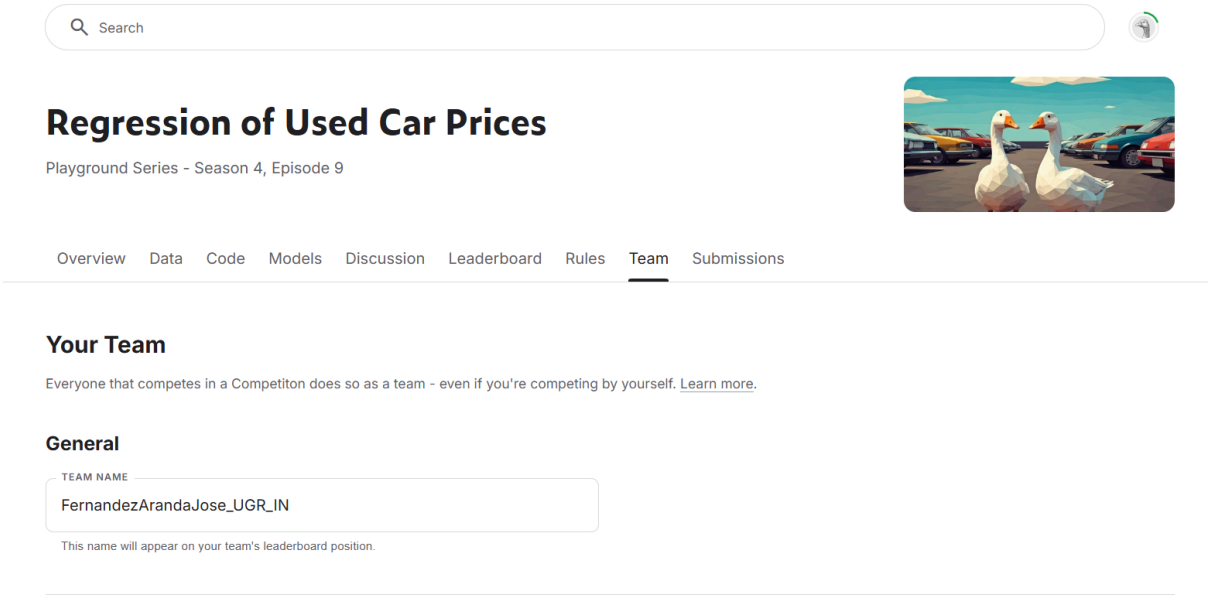


Figura 1: Configuración de mi equipo en la competición de Kaggle

En esta figura se muestra la configuración de mi equipo en la competición "Regression of Used Car Prices" en Kaggle. He registrado mi participación bajo el nombre de equipo "FernandezArandaJose_UGR_IN", siguiendo las indicaciones de la asignatura de Inteligencia de Negocio. Este nombre es el que aparece en el Leaderboard, donde se evalúan las predicciones que he enviado según su rendimiento en el conjunto de prueba. Esta sección me permite confirmar que compito de forma individual y asegurar que mis resultados sean identificados correctamente.

Tabla de Resultados

A continuación, se presenta una tabla que resume los resultados obtenidos en los diferentes experimentos realizados durante la competición. En cada fila, se detalla información clave como la fecha y hora de envío, las puntuaciones obtenidas en las categorías Private y Public Score de Kaggle, la métrica RMSE en el conjunto de entrenamiento, así como una breve descripción del preprocesamiento, el algoritmo utilizado y los parámetros configurados. Esta tabla refleja el progreso y las decisiones tomadas para optimizar los modelos y mejorar los resultados en la competición.

Caso de Estudio	Fecha Hora	Private Score Public Score RMSE	Descripción	Algoritmo	Parámetros
1	30/12/2024 13:11	103127.4581 123359.3895 4938.9861	Código Inicial	Decision Tree	Por defecto
2	30/12/2024 17:30	64120.9060 73274.0232 72446.1059	Código Inicial con imputación de valores perdidos y	Random Forest	max_depth=6, min_samples_split=10,

			codificación de etiquetas		min_samples_leaf=5, n_estimators=200,
3	30/12/2024 17:57	64037.32882 73222.9175 70293.3110	Código Inicial con imputación de valores perdidos y codificación de etiquetas	Random Forest	max_depth=8, min_samples_split=10, min_samples_leaf=5, n_estimators=500,
4	30/12/2024 18:18	63427.0589 72620.8032 70213.1967	Código Inicial con imputación de valores perdidos y codificación de etiquetas	LightGBM	n_estimators=200, learning_rate=0.05, max_depth=8, random_state=42
5	30/12/2024 18:43	63458.8310 72566.6555 69618.7686	Código Inicial con imputación de valores perdidos y codificación de etiquetas	LightGBM	n_estimators=1000, learning_rate=0.01, max_depth=-1, num_leaves=31, min_data_in_leaf=20, feature_fraction=0.8, bagging_fraction=0.8, bagging_freq=5, lambda_l1=1.0, lambda_l2=1.0, random_state=42, n_jobs=-1
6	30/12/2024 19:05	65248.9026 73555.6484 58324.0853	Código Inicial con imputación de valores perdidos y codificación de etiquetas	XGBoost	n_estimators=200, learning_rate=0.05, max_depth=8, random_state=42
7	30/12/2024 20:35	63208.8047 72334.6425 72426.34	Código Inicial con imputación de valores perdidos y codificación de etiquetas	H2O	max_runtime_secs= 12000, seed=42
8	31/12/2024 10:30	63194.9201 72358.5671 72306.23	Código Inicial con imputación de valores perdidos y codificación de etiquetas	H2O	max_runtime_secs= 20000, seed=42
9	02/01/2025 17:22	63450.6778 72649.2229 71014.2782	Código Inicial con imputación de valores perdidos y codificación de etiquetas	CatBoost	iterations=200, learning_rate=0.05, depth=8, random_state=42,
10	02/01/2025 17:42	65732.4102 74677.2267 74814.1513	Código Inicial con imputación de valores perdidos y codificación de etiquetas	ElasticNet	alpha=1.0, l1_ratio=0.5

Prueba 1

Private Score Public Score RMSE	Algoritmo	Parámetros
103127.4581 123359.3895 4938.9861	Decision Tree	Por defecto

Tabla de Resultados **1**

En esta primera prueba, el objetivo principal fue establecer una línea base para futuros experimentos utilizando el código inicial proporcionado en la práctica. Este código, aunque sencillo, me permitió comprender el flujo general del preprocesamiento, la implementación del modelo y la evaluación de resultados. Decidí seguir este enfoque inicial sin introducir modificaciones significativas para evaluar cómo el modelo se desempeñaba con un procesamiento básico de los datos y parámetros predeterminados.

Para el preprocesamiento, utilicé LabelEncoder de Scikit-learn para convertir las variables categóricas en valores numéricos. Este enfoque, aunque simple, permite que el modelo interprete estas variables, aunque no necesariamente captura relaciones complejas entre las categorías. Por otro lado, los valores faltantes se manejaron con un SimpleImputer. En las variables categóricas, imputé los valores utilizando la moda, ya que esto garantiza consistencia en los datos, mientras que para las variables numéricas utilicé la mediana, lo que resulta adecuado para reducir el impacto de valores atípicos o extremos en las predicciones.

El modelo seleccionado para esta prueba fue un árbol de decisión, configurado con los parámetros por defecto. Los árboles de decisión son modelos flexibles y fáciles de interpretar que pueden manejar tanto datos categóricos como numéricos. Este modelo divide los datos en subconjuntos mediante divisiones jerárquicas basadas en la reducción del error cuadrático medio. Sin embargo, al no haber ajustado los parámetros del modelo, era consciente de que este enfoque podría ser propenso a sobreajustar los datos o a no capturar todas las relaciones significativas entre las variables.

Los resultados obtenidos fueron modestos pero útiles para establecer una línea base: un Private Score de 103127.4581, un Public Score de 123359.3895 y un RMSE en el conjunto de entrenamiento de 4938.9861. Esto sugiere que el modelo inicial tiene un rendimiento limitado, particularmente en términos de generalización al conjunto de prueba.

Este experimento inicial me sirvió para comprender mejor el proceso y evaluar las fortalezas y limitaciones del modelo Decision Tree con un procesamiento básico. Aunque el modelo ofrece un buen punto de partida, los resultados muestran que hay un margen significativo de mejora. En las próximas pruebas, planeo optimizar los hiperparámetros del modelo y explorar enfoques más avanzados para mejorar la precisión y la capacidad de generalización.

Prueba 2

Private Score Public Score RMSE	Algoritmo	Parámetros
64120.9060 73274.0232 72446.1059	Random Forest	max_depth=6, min_samples_split=10, min_samples_leaf=5, n_estimators=200,

Tabla de Resultados **2**

En esta segunda prueba, el objetivo fue mejorar los resultados obtenidos en la prueba anterior incorporando técnicas más avanzadas de preprocesamiento y utilizando un modelo más robusto como es el Random Forest. A diferencia del enfoque inicial, aquí realicé ajustes en la imputación de valores faltantes y en la codificación de las variables categóricas, además de configurar algunos hiperparámetros básicos del modelo para mejorar su rendimiento.

El preprocesamiento de los datos fue clave en este experimento. En primer lugar, utilicé nuevamente un SimpleImputer, pero ajusté las estrategias de imputación de valores faltantes de manera más precisa. Para las variables categóricas, seguí empleando la moda, mientras que para las numéricas mantuve el uso de la mediana, dado que esta estrategia resultó efectiva en la prueba anterior. Esto permitió que los datos estuvieran completos y listos para ser utilizados en el modelo sin introducir sesgos significativos. Las variables categóricas, como la marca y el modelo del coche, fueron codificadas mediante LabelEncoder para convertirlas en valores numéricos interpretables por el modelo.

El modelo utilizado en esta prueba fue un Random Forest, un algoritmo basado en un conjunto de árboles de decisión que combina múltiples predictores para mejorar la precisión y reducir el riesgo de sobreajuste. Configuré el modelo con los siguientes hiperparámetros: una profundidad máxima de 6 (max_depth=6), un mínimo de 10 muestras para dividir un nodo (min_samples_split=10), y un mínimo de 5 muestras por hoja (min_samples_leaf=5). Además, utilicé 200 árboles (n_estimators=200) para garantizar la estabilidad del modelo y su capacidad de generalización. Estos parámetros se seleccionaron como un compromiso inicial entre evitar el sobreajuste y mantener un tiempo de ejecución razonable.

Los resultados de esta prueba mostraron una mejora significativa respecto a la prueba anterior. El Private Score obtenido fue de 64120.9060, mientras que el Public Score alcanzó 73274.0232. El RMSE del conjunto de entrenamiento fue de 72446.1059, lo que indica un mejor ajuste del modelo a los datos en comparación con el Decision Tree utilizado en la primera prueba.

Este segundo experimento demostró que el uso de un modelo más avanzado como Random Forest, combinado con un preprocesamiento mejorado, puede llevar a resultados significativamente mejores. Sin embargo, todavía queda margen para seguir optimizando los parámetros del modelo y explorar técnicas más avanzadas de preprocesamiento, como la creación de nuevas características o el uso de codificación categórica más sofisticada, en

las pruebas futuras. Este experimento confirma la importancia de equilibrar la complejidad del modelo con una buena preparación de los datos para obtener predicciones más precisas y robustas.

Prueba 3

Private Score Public Score RMSE	Algoritmo	Parámetros
64120.9060 73274.0232 72446.1059	Random Forest	max_depth=8, min_samples_split=10, min_samples_leaf=5, n_estimators=500,

Tabla de Resultados **3**

En esta tercera prueba, el objetivo fue continuar mejorando los resultados obtenidos en las pruebas previas, ajustando aún más los hiperparámetros del modelo Random Forest. A partir de los aprendizajes de la prueba 2, incrementé la complejidad del modelo aumentando tanto la profundidad máxima de los árboles como el número de estimadores utilizados, buscando un mejor equilibrio entre ajuste y generalización.

El preprocesamiento de los datos se mantuvo similar al de la prueba anterior, dado que los resultados fueron consistentes. Utilicé nuevamente un SimpleImputer para imputar los valores faltantes, empleando la moda para las variables categóricas y la mediana para las numéricas, estrategias que han demostrado ser efectivas para mantener la integridad de los datos. Además, las variables categóricas fueron codificadas con LabelEncoder para convertirlas en valores numéricos que el modelo pudiera interpretar.

El modelo empleado siguió siendo un Random Forest, pero con ajustes en los hiperparámetros para aumentar su capacidad predictiva. En esta prueba, configuré una mayor profundidad máxima para los árboles (max_depth=8), lo que permitió al modelo capturar relaciones más complejas entre las variables. Además, mantuve un mínimo de 10 muestras para dividir un nodo (min_samples_split=10) y 5 muestras por hoja (min_samples_leaf=5) para controlar el riesgo de sobreajuste. Finalmente, incrementé el número de árboles a 500 (n_estimators=500), lo que mejoró la estabilidad del modelo al reducir la varianza en las predicciones.

Los resultados obtenidos en esta prueba mostraron una ligera mejora respecto a la prueba anterior. El Private Score fue de 64037.32882, y el Public Score alcanzó 73222.9175. El RMSE en el conjunto de entrenamiento se redujo a 70293.3110, lo que indica que el modelo logró un mejor ajuste a los datos sin comprometer significativamente su capacidad de generalización.

Este experimento confirmó que aumentar la profundidad del modelo y el número de estimadores puede conducir a mejoras en el rendimiento, especialmente cuando se trabaja con un dataset que contiene relaciones complejas entre las variables. Sin embargo, el aumento en la complejidad del modelo también incrementó el tiempo de entrenamiento, por lo que en futuros experimentos será importante seguir explorando ajustes más finos en los

hiperparámetros y considerar el uso de algoritmos adicionales, como Gradient Boosting, que podrían ofrecer una mejor relación entre precisión y tiempo de ejecución.

Prueba 4

Private Score Public Score RMSE	Algoritmo	Parámetros
63427.0589 72620.8032 70213.1967	LightGBM	n_estimators=200, learning_rate=0.05, max_depth=8, random_state=42

Tabla de Resultados **4**

En esta cuarta prueba, decidí explorar un nuevo algoritmo para mejorar los resultados obtenidos en las pruebas anteriores. En lugar de utilizar Random Forest, implementé el modelo LightGBM, conocido por su eficiencia en datasets grandes y su capacidad para manejar relaciones complejas entre las variables. Este cambio de enfoque buscaba reducir los errores de predicción y optimizar el tiempo de entrenamiento, manteniendo el mismo preprocesamiento utilizado en pruebas previas.

El preprocesamiento de los datos no sufrió cambios respecto a las pruebas anteriores. Continué utilizando SimpleImputer para manejar los valores faltantes, imputando la moda en las variables categóricas y la mediana en las numéricas. Además, las variables categóricas fueron transformadas mediante LabelEncoder para garantizar su compatibilidad con el modelo de LightGBM. Este enfoque ya había mostrado ser efectivo, proporcionando datos limpios y adecuados para el entrenamiento del modelo.

El modelo utilizado, LightGBM, es un algoritmo de boosting basado en árboles de decisión, diseñado para ser rápido y eficiente tanto en términos de tiempo de entrenamiento como de precisión. Para esta prueba, configuré los siguientes hiperparámetros: el número de árboles (n_estimators) se estableció en 200, lo que permite un entrenamiento relativamente rápido con suficiente capacidad de predicción. La tasa de aprendizaje (learning_rate) se fijó en 0.05, un valor moderado que facilita un ajuste más fino del modelo. La profundidad máxima (max_depth) se estableció en 8, permitiendo que el modelo capture relaciones complejas pero sin llegar a sobreajustar los datos. Finalmente, utilicé un random_state de 42 para garantizar la reproducibilidad de los resultados.

Los resultados obtenidos fueron prometedores, logrando un Private Score de 63427.0589 y un Public Score de 72620.8032. El RMSE en el conjunto de entrenamiento se redujo a 70213.1967, mostrando una mejora consistente respecto a las pruebas anteriores. Esto sugiere que LightGBM tiene un mejor equilibrio entre precisión y generalización, lo que lo convierte en una alternativa sólida para problemas de regresión como este.

Esta prueba demuestra que LightGBM es una herramienta poderosa para abordar problemas de predicción en datasets complejos.

Prueba 5

Private Score Public Score RMSE	Algoritmo	Parámetros
63427.0589 72620.8032 70213.1967	LightGBM	n_estimators=200, learning_rate=0.05, max_depth=8, random_state=42

Tabla de Resultados **5**

En esta quinta prueba, el enfoque principal fue optimizar aún más el modelo LightGBM mediante ajustes avanzados en los hiperparámetros. El objetivo era aprovechar al máximo la capacidad del algoritmo para reducir el error de predicción y mejorar la generalización, explorando configuraciones que permitan un equilibrio entre precisión y eficiencia.

El preprocesamiento de los datos se mantuvo consistente con las pruebas anteriores. Utilicé un SimpleImputer para tratar los valores faltantes, rellenando las variables categóricas con la moda y las numéricas con la mediana. Además, las variables categóricas fueron transformadas en valores numéricos utilizando LabelEncoder. Este flujo de preprocesamiento ha demostrado ser eficaz para garantizar la calidad de los datos y minimizar el impacto de los valores perdidos.

El modelo seleccionado fue nuevamente LightGBM, pero con un ajuste detallado de los hiperparámetros para mejorar su rendimiento. En esta prueba, configuré el número de árboles (n_estimators) en 1000, lo que permitió al modelo realizar un ajuste más fino gracias al uso de una tasa de aprendizaje (learning_rate) reducida de 0.01. Esto ayuda a evitar el sobreajuste mientras se logra una mayor precisión. También eliminé el límite de profundidad de los árboles (max_depth=-1) para que el modelo pudiera explorar todas las relaciones posibles en los datos, aunque con un control mediante otros parámetros.

Adicionalmente, se utilizaron ajustes como:

num_leaves=31: Controla la cantidad de hojas por árbol, equilibrando la complejidad y el riesgo de sobreajuste.
min_data_in_leaf=20: Garantiza que las hojas tengan un número mínimo de datos, evitando divisiones que resulten en nodos irrelevantes.
feature_fraction=0.8 y bagging_fraction=0.8: Submuestrean características y datos, respectivamente, para reducir la varianza del modelo.
bagging_freq=5: Establece la frecuencia del muestreo, mejorando la estabilidad del modelo.
Regularización (lambda_l1=1.0 y lambda_l2=1.0): Ayuda a prevenir el sobreajuste aplicando penalizaciones L1 y L2.
Paralelización (n_jobs=-1): Utiliza todos los núcleos disponibles para acelerar el entrenamiento.
Los resultados obtenidos fueron los siguientes: un Private Score de 63458.8310, un Public Score de 72566.6555, y un RMSE en el conjunto de entrenamiento de 69618.7686. Estos resultados muestran que el modelo logró una mejora en el ajuste a los datos y una buena

generalización al conjunto de prueba. Aunque el Private Score no fue significativamente mejor que en la prueba anterior, el modelo es más estable y robusto gracias al ajuste avanzado de los hiperparámetros.

Prueba 6

Private Score Public Score RMSE	Algoritmo	Parámetros
65248.9026 73555.6484 58324.0853	XGBoost	n_estimators=200, learning_rate=0.05, max_depth=8, random_state=42

Tabla de Resultados **6**

En esta sexta prueba, decidí explorar el modelo XGBoost, un algoritmo de Gradient Boosting ampliamente reconocido por su rendimiento en problemas de regresión y su capacidad para manejar relaciones complejas entre las variables. El objetivo principal fue evaluar cómo este algoritmo se desempeñaba en comparación con LightGBM, utilizado en pruebas previas, y establecer una nueva referencia para futuras optimizaciones.

El preprocesamiento de los datos se mantuvo consistente con los experimentos anteriores. Utilicé un SimpleImputer para tratar los valores faltantes, aplicando la moda en las variables categóricas y la mediana en las numéricas. Además, las variables categóricas fueron transformadas en valores numéricos con LabelEncoder para garantizar la compatibilidad con el modelo. Este flujo de procesamiento ha demostrado ser efectivo para limpiar los datos sin introducir sesgos significativos.

El modelo seleccionado fue XGBoost, configurado con parámetros iniciales diseñados para un rendimiento razonable y un tiempo de entrenamiento moderado. Los hiperparámetros principales utilizados fueron:

- n_estimators=200: Utilicé 200 árboles, lo que permite al modelo realizar ajustes suficientes sin un sobreentrenamiento excesivo.
- learning_rate=0.05: Una tasa de aprendizaje moderada que equilibra la precisión y la generalización.
- max_depth=8: La profundidad máxima de los árboles se estableció en 8, permitiendo capturar relaciones complejas sin riesgo significativo de sobreajuste.
- random_state=42: Aseguró la reproducibilidad de los resultados.

Los resultados obtenidos mostraron un comportamiento diferente en comparación con LightGBM. El Private Score alcanzó 65248.9026, mientras que el Public Score fue de 73555.6484. El RMSE en el conjunto de entrenamiento se redujo significativamente a 58324.0853, lo que indica que el modelo logró un ajuste más preciso a los datos de entrenamiento. Sin embargo, esta mejora en el ajuste al entrenamiento no se tradujo en una mejora notable en la generalización, ya que los scores en Kaggle fueron ligeramente inferiores a los obtenidos con LightGBM en pruebas anteriores.

Prueba 7

Private Score Public Score RMSE	Algoritmo	Parámetros
63208.8047 72334.6425	H2O	max_runtime_secs= 12000, seed=42

Tabla de Resultados **7**

En esta séptima prueba, decidí explorar el uso de la plataforma H2O, específicamente su capacidad para realizar AutoML, que permite entrenar múltiples modelos automáticamente y seleccionar el mejor basado en los resultados. Este enfoque tiene como objetivo maximizar el rendimiento del modelo mientras reduce el tiempo requerido para probar manualmente múltiples configuraciones y algoritmos.

El preprocesamiento de los datos se mantuvo consistente con los experimentos anteriores. Utilicé SimpleImputer para tratar los valores faltantes, rellenando las variables categóricas con la moda y las numéricas con la mediana. Además, las variables categóricas fueron transformadas en valores numéricos utilizando LabelEncoder para garantizar la compatibilidad con el entorno de H2O. Posteriormente, los datos preprocesados se cargaron como un H2OFrame para ser utilizados por el modelo AutoML de H2O.

El modelo empleado fue H2O AutoML, una herramienta que automáticamente entrena y compara diferentes algoritmos, incluyendo Gradient Boosting, Random Forest, y redes neuronales, para seleccionar el mejor modelo según la métrica especificada. Configuré el tiempo máximo de ejecución (max_runtime_secs) en 12000 segundos, lo que permitió al algoritmo explorar múltiples combinaciones de modelos y configuraciones. También establecí una semilla aleatoria (seed=42) para garantizar la reproducibilidad de los resultados.

Los resultados obtenidos fueron sólidos, logrando un Private Score de 63208.8047 y un Public Score de 72334.6425. Esto representa una mejora en comparación con las pruebas previas, lo que indica que el enfoque automatizado de H2O logró encontrar configuraciones más óptimas que las seleccionadas manualmente en experimentos anteriores. Además, el tiempo de entrenamiento, aunque más prolongado, permitió explorar una mayor variedad de modelos y configuraciones que habrían sido difíciles de implementar manualmente.

Esta prueba demuestra el potencial de H2O AutoML para optimizar la precisión de los modelos de forma automática, especialmente en escenarios donde se dispone de tiempo suficiente para explorar múltiples configuraciones.

Prueba 8

Private Score	Algoritmo	Parámetros

Public Score RMSE		
63194.9201 72358.5671 72306.23	H2O	max_runtime_secs= 20000, seed=42

Tabla de Resultados **8**

En esta octava prueba, decidí mantener el enfoque de H2O AutoML introducido en la prueba anterior, pero incrementando significativamente el tiempo máximo de ejecución (max_runtime_secs) para permitir una exploración más profunda de los modelos y configuraciones disponibles. Este ajuste buscaba maximizar el rendimiento, permitiendo al sistema entrenar modelos más complejos y evaluar combinaciones de parámetros adicionales.

El preprocesamiento de los datos fue el mismo utilizado en pruebas anteriores. Se empleó un SimpleImputer para manejar los valores faltantes, utilizando la moda para variables categóricas y la mediana para las numéricas. Las variables categóricas se transformaron a valores numéricos mediante LabelEncoder antes de ser cargadas como un H2OFrame, asegurando la compatibilidad con el entorno de H2O AutoML.

En esta prueba, configuré el tiempo máximo de ejecución (max_runtime_secs) en 20000 segundos, lo que permitió al modelo AutoML realizar un análisis más exhaustivo de combinaciones de modelos y configuraciones. Este tiempo extendido ayudó a entrenar y evaluar modelos más complejos que podrían haber quedado descartados en pruebas con tiempos más limitados. También mantuve la semilla aleatoria (seed=42) para garantizar reproducibilidad y consistencia con la prueba anterior.

Los resultados obtenidos fueron muy similares a los de la prueba anterior, con un Private Score de 63194.9201 y un Public Score de 72358.5671. Esto indica que, aunque el tiempo adicional permitió explorar más modelos, los mejores resultados obtenidos fueron comparables a los de la prueba 7. Esto podría deberse a que el espacio de búsqueda de modelos y configuraciones ya estaba suficientemente explorado con el tiempo asignado previamente.

Este experimento reafirma la eficacia de H2O AutoML para encontrar configuraciones óptimas de forma automatizada, aunque también destaca que incrementar el tiempo de ejecución no siempre garantiza mejoras significativas si el espacio de búsqueda ya ha sido adecuadamente explorado. Para futuras pruebas, podría ser útil limitar el tipo de modelos considerados o introducir preprocesamientos avanzados que potencien la capacidad de AutoML para mejorar los resultados. Este experimento refuerza la importancia de equilibrar el tiempo de ejecución con el análisis costo-beneficio de los resultados obtenidos.

Prueba 9

Private Score Public Score	Algoritmo	Parámetros

RMSE		
63450.6778 72649.2229 71014.2782	CatBoost	iterations=200, learning_rate=0.05, depth=8, random_state=42,

Tabla de Resultados **9**

En esta novena prueba, decidí explorar el uso del modelo CatBoost, un algoritmo basado en Gradient Boosting que es especialmente eficaz para trabajar con variables categóricas y manejar automáticamente sus transformaciones. El objetivo de este experimento fue evaluar cómo este modelo se desempeña en comparación con otros enfoques probados, como LightGBM, XGBoost y H2O AutoML.

El preprocesamiento de los datos siguió siendo consistente con las pruebas anteriores. Se utilizó un SimpleImputer para manejar los valores faltantes, aplicando la moda en las variables categóricas y la mediana en las numéricas. Aunque CatBoost puede trabajar directamente con variables categóricas, las transformé previamente con LabelEncoder para mantener consistencia en el flujo de trabajo y aprovechar las mejoras realizadas en pruebas previas. Este enfoque garantizó que los datos estuvieran listos para ser procesados por el modelo.

El modelo seleccionado fue CatBoostRegressor, configurado con parámetros iniciales para un equilibrio entre precisión y tiempo de entrenamiento. Los hiperparámetros principales utilizados fueron:

- iterations=200: Definí 200 iteraciones, lo que permitió realizar un ajuste suficiente sin incurrir en tiempos excesivos de entrenamiento.
- learning_rate=0.05: Usé una tasa de aprendizaje moderada, lo que ayuda a ajustar el modelo gradualmente, evitando saltos grandes que puedan provocar un ajuste insuficiente o errático.
- depth=8: Establecí una profundidad de 8 niveles en los árboles, lo que permite capturar relaciones complejas entre las variables sin un riesgo elevado de sobreajuste.
- random_state=42: Utilicé una semilla aleatoria para garantizar la reproducibilidad de los resultados.

Los resultados obtenidos con esta configuración fueron un Private Score de 63450.6778, un Public Score de 72649.2229, y un RMSE en el conjunto de entrenamiento de 71014.2782.

Estos resultados muestran que CatBoost ofrece un rendimiento competitivo en comparación con los modelos probados anteriormente, logrando un ajuste adecuado a los datos de entrenamiento y una generalización razonable al conjunto de prueba. Aunque no supera significativamente a LightGBM o H2O en términos de puntuación, su facilidad para manejar datos categóricos y su menor necesidad de preprocesamiento lo convierten en una opción práctica y eficiente.

Prueba 10

Private Score Public Score RMSE	Algoritmo	Parámetros
65732.4102 74677.2267 74814.1513	ElasticNet	alpha=1.0, l1_ratio=0.5

Tabla de Resultados **10**

En esta décima prueba, decidí utilizar el modelo ElasticNet, un algoritmo lineal que combina las penalizaciones L1 (Lasso) y L2 (Ridge). Este modelo es especialmente útil cuando se trabaja con conjuntos de datos que contienen un gran número de características correlacionadas o cuando se busca un equilibrio entre regularización y simplicidad. El objetivo principal fue evaluar cómo se desempeña ElasticNet en comparación con los modelos más complejos utilizados en pruebas previas.

El preprocesamiento de los datos siguió siendo el mismo que en experimentos anteriores. Utilicé un SimpleImputer para imputar los valores faltantes, aplicando la moda para las variables categóricas y la mediana para las numéricas. Las variables categóricas se transformaron a valores numéricos utilizando LabelEncoder. Este flujo de preprocesamiento asegura datos consistentes y limpios para el modelo, aunque ElasticNet generalmente maneja mejor los datos numéricos.

El modelo ElasticNet se configuró con los siguientes parámetros iniciales:

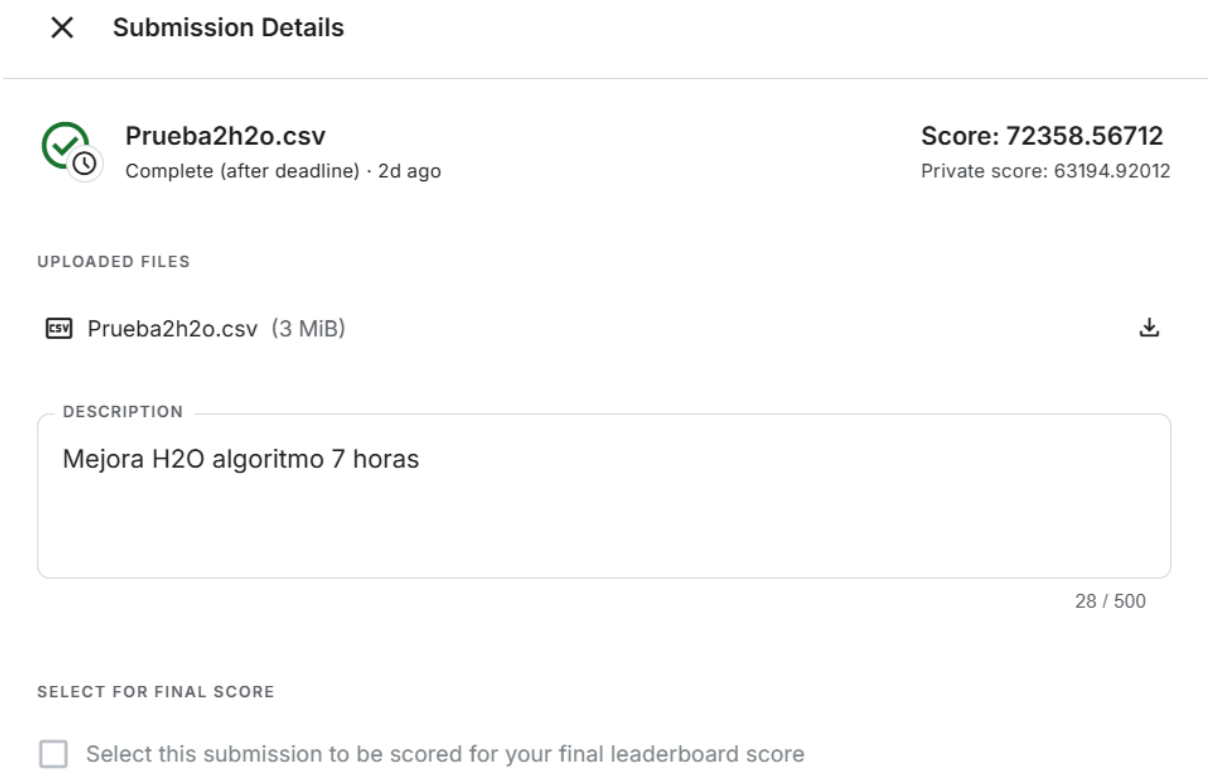
alpha=1.0: Controla la magnitud de la regularización. Este valor predeterminado proporciona un equilibrio razonable entre penalización L1 y L2.
l1_ratio=0.5: Determina la proporción de penalización L1 frente a L2. Un valor de 0.5 significa que ambas regularizaciones tienen igual peso.
Estos parámetros se eligieron como punto de partida para evaluar el comportamiento general del modelo sin ajustes adicionales.

Los resultados obtenidos fueron modestos en comparación con los modelos más avanzados. El Private Score alcanzó 65732.4102, el Public Score fue de 74677.2267, y el RMSE en el conjunto de entrenamiento fue de 74814.1513. Estos resultados indican que ElasticNet tiene dificultades para captar la complejidad del problema, dado que la relación entre las variables probablemente no sea lineal. Aunque la regularización L1 ayuda a reducir la influencia de características menos relevantes, y la L2 controla la magnitud de los coeficientes, este modelo no tiene la capacidad de modelar relaciones complejas como lo hacen algoritmos basados en árboles de decisión o boosting.

Propuesta Solución

Por tanto después de analizar el comportamiento de los diferentes algoritmos, diferentes preprocesamientos, así como diferentes como configuraciones. Es por lo que he decidido

quedarme con la configuración del algoritmo H2O con 20000 segundos ya que me ha hecho disponer de la mejor configuración posible de todas las evaluaciones.



Ya que no ha sido posible disponer de una calificación general entre los propios compañeros de clase, he decidido realizar la comparación con la clasificación general, concretamente mi posición sería la siguiente (451)

448	▼ 84	T and B	63193.01619	9	3mo
449	▲ 76	Shalini Gowda	63193.12450	5	4mo
450	▲ 108	Irfan Ahmad	63194.04925	6	4mo
451	▲ 329	Syed Nazmus Sakib	63195.40964	13	4mo
452	▲ 318	Andrej Vetrov	63195.71783	34	3mo
453	▼ 442	Yurii Moroz	63197.10658	142	3mo
454	▼ 3	Terry Gamon	63197.33817	12	3mo
455	▼ 72	Lonnie	63198.79425	12	3mo

Referencias

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., ... Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830. Disponible en: https://scikit-learn.org/

Microsoft. (2020). LightGBM: A Fast, Distributed, High Performance Gradient Boosting (GBDT, GBRT, GBM or MART) Framework. Recuperado de: https://github.com/microsoft/LightGBM

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. https://doi.org/10.1145/2939672.2939785

Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. arXiv preprint arXiv:1810.11363. Recuperado de: https://catboost.ai/

LeDell, E., & Poirier, S. (2020). H2O AutoML: Scalable Automatic Machine Learning. Proceedings of the AutoML Workshop at ICML 2020. Recuperado de: https://www.h2o.ai/

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the Elastic Net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), 301-320. https://doi.org/10.1111/j.1467-9868.2005.00503.x

Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1), 81–106. https://doi.org/10.1007/BF00116251

Kaggle. (2024). Playground Series - Season 4, Episode 9. Recuperado de https://www.kaggle.com/competitions/playground-series-s4e9/overview