Examen Técnico

Autor:
Victor Manuel Fonte Chavez
Agosto 2024

1. Metodología

La metodología empleada se dividió en varias etapas críticas:

- **Preparación de Datos:** Carga y limpieza de un conjunto de datos extenso que contiene características y precios de decenas de miles de diamantes. Este paso incluyó la corrección de errores y la imputación de valores faltantes.
- **Análisis Exploratorio de Datos (EDA):** Exploración de las características de los diamantes, como carat, corte, color, y claridad, para entender su distribución y relación con el precio.
- **Modelización:** Desarrollo de modelos predictivos utilizando técnicas como regresión lineal, bosques aleatorios y máquinas de soporte vectorial para estimar el valor de los diamantes basados en sus características.
- **Validación del Modelo:** Evaluación de los modelos mediante técnicas de validación cruzada y ajuste basado en el conjunto de datos de prueba para garantizar la generalización y precisión de las predicciones.

2. Limpieza de los Datos

2.1 Visualización de Valores Perdidos

Se iniciara el análisis dando un vistazo general a los datos para visualizar posibles errores. Primeramente buscamos columnas sin sentido o duplicadas para eliminarlas. En este caso, la columna *Unnamed: 0* se eliminó ya que parecía ser una columna de IDs repetida. Además, se corrigió un problema en los datos de coordenadas donde el nombre de la columna *'longitude '* contenía un espacio adicional. Esta columna fue renombrada a *longitude* para evitar problemas futuros.

Posteriormente, se combinaron los datos limpios con las coordenadas corregidas en un solo DataFrame. El DataFrame final contiene 53930 filas y 12 columnas, con información detallada sobre las características de los diamantes, incluyendo su *carat*, *cut*, *color*, *clarity*, *depth*, *table*, *price*, y dimensiones (x, y, z), así como las coordenadas de latitud y longitud.

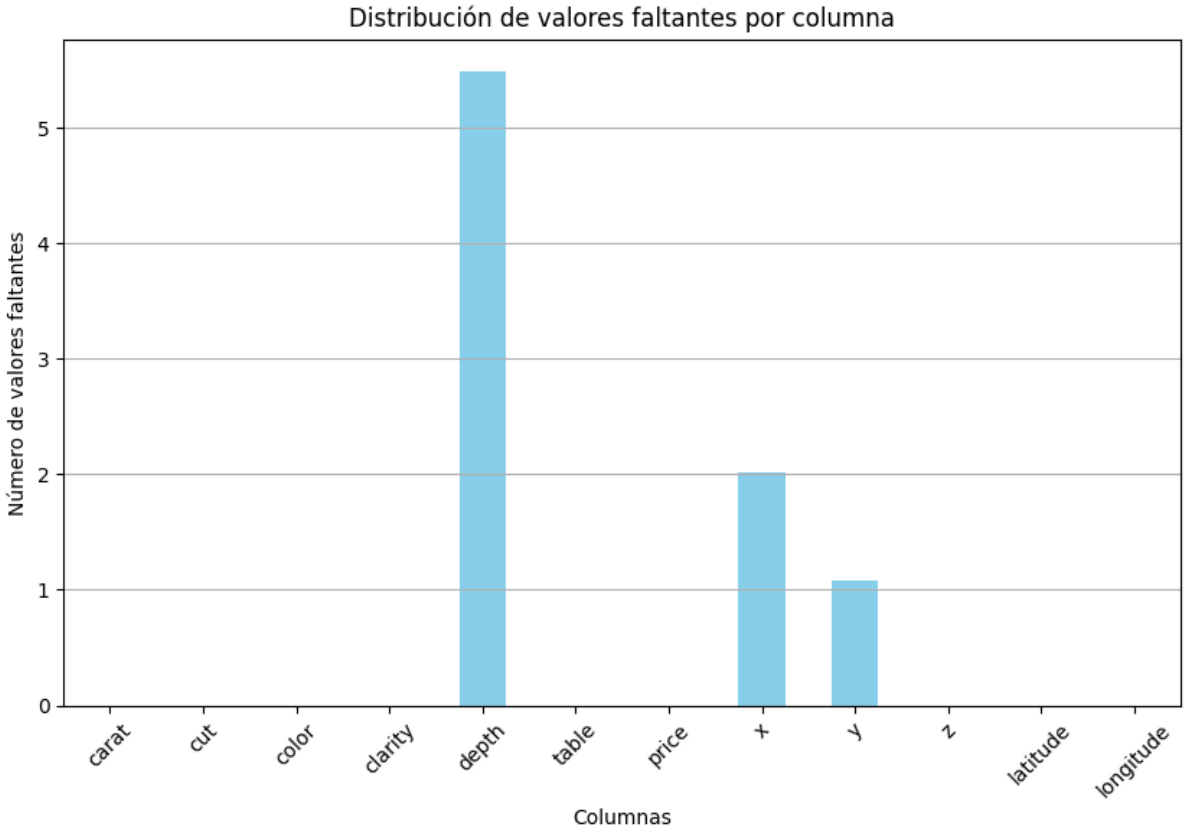

Figura 1: Cantidad de valores ausentes por variable

En esta sección, se realizó una visualización de la cantidad de datos faltantes por columna. Utilizando una gráfica de barras, se pudo identificar las columnas con valores ausentes y la proporción de estos respecto al total de datos.

La gráfica resultante muestra que las columnas *depth*, *x, y* tienen valores faltantes significativos. Esto permite enfocar los esfuerzos de limpieza y corrección en estas columnas para asegurar la integridad del análisis posterior.

En esta sección, se generó un mapa de calor para visualizar la distribución de los valores faltantes en el dataset. La visualización se realizó dividiendo el dataset en cinco partes para facilitar el análisis y acelerar el proceso de graficado.

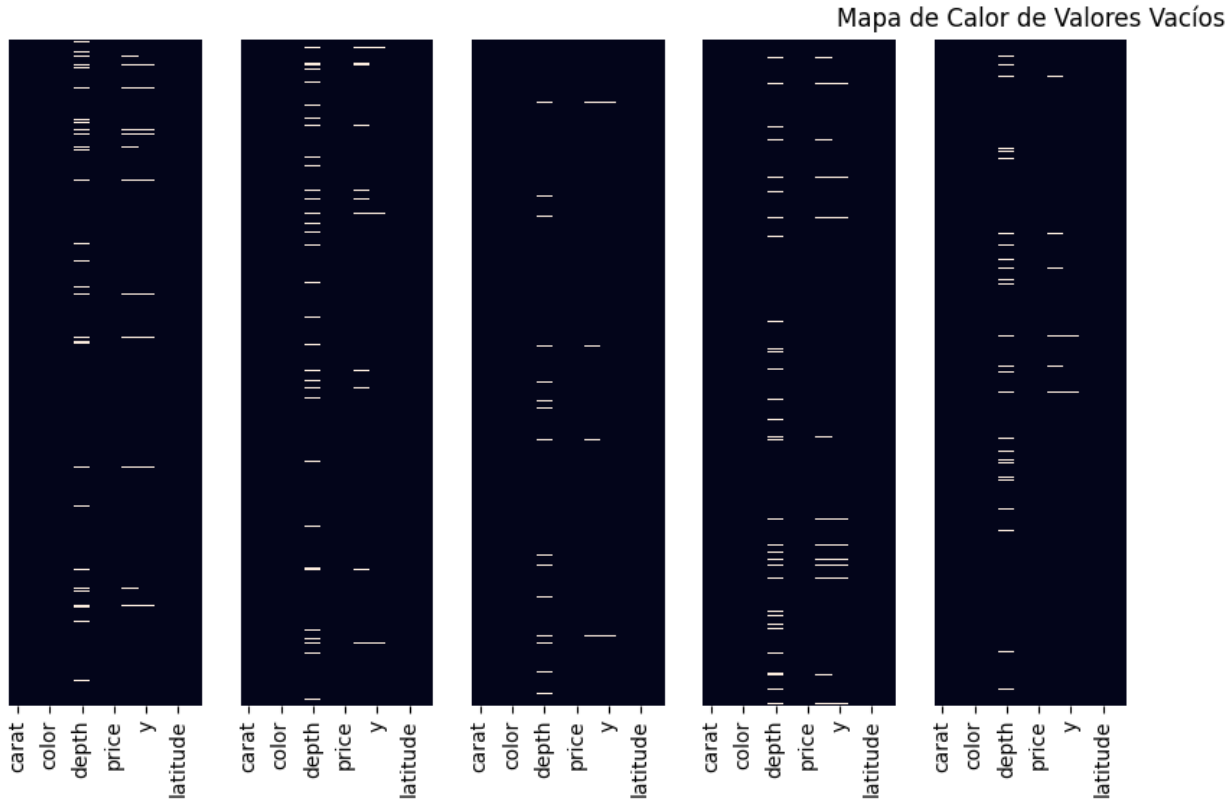

Figura 2: Distribucion de valores ausentes por variable

El mapa de calor muestra la presencia de valores faltantes en diferentes partes del dataset, destacando las áreas con datos ausentes mediante bloques blancos en el gráfico. Esta visualización permite identificar fácilmente las secciones del dataset que requieren atención en términos de imputación o limpieza de datos, además de dar la idea de cuan aleatorio es la generación de los valores perdidos en las variables, importante para los procesos de imputación

2.2 Imputación de Valores Perdidos

En esta sección, se realizó un análisis más detallado de los valores faltantes en el dataset. Como se observó en la gráfica de barras y en el mapa de calor, la columna con mayor cantidad de valores faltantes es *clarity* con un 0.05 % de valores faltantes. Además, los valores faltantes están distribuidos de manera no uniforme, lo que sugiere que podría ser necesario eliminar filas con valores faltantes o implementar una imputación de valores faltantes con MICE (Multiple Imputation by Chained Equations).

Para entender mejor el esquema de valores ausentes (MCAR, MAR, MNAR), se realizó un test de correlaciones de Pearson entre columnas de valores faltantes binarizadas, esto para demostrar que el mecanismo generador de valores faltantes en cada variable es independiente de entre todas las demás variables con valores ausentes. Se realizo un test de correlaciones de Pearson, del cual se tiene que rechazar la hipótesis nula (p-valor <0.05) para que se pueda realizar imputación sobre los valores perdidos. Se debe tener en cuenta también que se están realizando varios test por lo que es necesario controlar la tasa de falsos descubrimientos. En este algoritmo, primero se ordenan todos los p-valores obtenidos de las pruebas estadísticas en orden ascendente.Para cada p-valor ordenado, se calcula el p-valor ajustado utilizando la fórmula:

$$p_i^{\text{ajustado}} = \frac{p_i \cdot N}{i}$$

donde p_i es el i-ésimo p-valor ordenado, N es el número total de pruebas, y i es la posición del p-valor en la lista ordenada. Los resultados del test se pueden ver a continuación

	depth	x	y
depth	0.0	0.0	0.0
x	0.0	0.0	0.0
y	0.0	0.0	0.0

Como se puede ver se rechaza la hipótesis nula de que están correlacionados entre si por lo que se puede proceder a imputar los valores faltantes con un modelo MICE (Multiple Imputation by Chained Equations).

El algoritmo MICE es una técnica sofisticada que se basa en la creación de múltiples imputaciones para cada valor faltante. Este método es útil porque permite mantener la incertidumbre sobre los valores imputados reflejando las

variaciones naturales en los datos.

2.3 Limpieza de variables cualitativas

En esta sección, se aborda la limpieza de variables cualitativas para asegurar que los datos estén libres de errores y sean consistentes. El objetivo principal es identificar y corregir cualquier problema en las variables cualitativas, como caracteres especiales o datos mal formateados, que podrían afectar el análisis posterior.

Primeramente se identifican las columnas del DataFrame que contienen datos cualitativos (de tipo object). En este caso, las columnas son *cut*, *color*, *clarity* y *latitude*. Sin embargo se detecta que la columna *latitude* tiene un problema donde algunos valores contienen caracteres no numéricos. Se utiliza una expresión regular `r'[a-zA-Z]'` para identificar filas donde *latitude* contiene letras. El resultado muestra que en la posición 48185, el valor es '33q.200088', lo que claramente es un error, ya que 'latitude' debería contener solo valores numéricos. Debido a que el valor en la posición 48185 está claramente mal formateado y no se puede determinar su valor correcto, se opta por eliminar esta fila. Después de eliminar la fila problemática, se convierte la columna *latitude* a tipo float para asegurar que todos los valores restantes sean numéricos.

Ahora validaremos que las variables cualitativas (*cut*, *color*, y *clarity*) contengan únicamente valores aceptables y consistentes, y eliminar cualquier inconsistencia encontrada. Se define una función `validate_cat_vars` que toma el DataFrame data como entrada y verifica cada columna cualitativa contra un conjunto de valores válidos predefinidos. Las inconsistencias se almacenan en un diccionario. Al correr este codigo se pueden observar que hay 74 filas con errores. Pero si nos nos fijamos mejor se puede apreciar que estos errores son principalmente sustituciones de letras con caracteres especiales como por ejemplo (Very Goo!d, #SI!1, &VS2, ...). Por lo tanto, otra vez se itera por todas las columnas cualitativas (de tipo object) y se eliminan símbolos especiales utilizando una expresión regular `r'[^ \w\s]'` que mantiene solo caracteres alfanuméricos y espacios.

2.4 Limpieza de variables cuantitativas

En esta sección, se realiza una búsqueda de valores negativos o nulos en el

dataset, ya que se espera que todos los valores sean positivos para mantener la coherencia y validez de los datos. El objetivo es identificar y eliminar cualquier fila que contenga valores negativos o nulos en las columnas relevantes del dataset.

Se define una función `get_inconsistencies` que devuelve las filas del dataset donde alguna de las columnas *carat*, *depth*, *table*, *price*, x, y o z contienen valores negativos o nulos, exactamente 35 filas contienen. Para resolver esto, ya no son muchas, optamos por eliminarlas.

3. Análisis Exploratorio

3.1 Análisis descriptivo Variables Cuantitativas

Antes que nada visualizaremos la distribución de la variable objetivo *price* para evaluar si es necesario aplicar alguna normalización o transformación. Esto se hace para entender mejor la distribución de los datos y potencialmente mejorar la efectividad de los modelos predictivos que se puedan usar posteriormente.

En la figura 3 se pueden apreciar los resultados. Las gráficas generadas muestran las distribuciones de la variable *price* en su forma original, después de aplicar una transformación de raíz cuadrada y después de aplicar una transformación logarítmica.

Distribución Original: La distribución del precio muestra una fuerte asimetría hacia la derecha con una concentración de datos en valores bajos y algunos valores extremadamente altos.

Transformación de Raíz Cuadrada: Esta transformación reduce algo de la asimetría, pero aún se observan valores altos desplazados hacia la derecha.

Transformación Logarítmica: Esta transformación logra una distribución más simétrica y normalizada, lo cual puede ser útil para aplicar modelos que asumen una distribución normal de los datos.

Ahora realizaremos un análisis exploratorio de las relaciones entre las variables explicativas. El objetivo es calcular y visualizar las correlaciones entre las variables cuantitativas del dataset. Esto ayuda a entender las relaciones lineales entre las variables, lo cual puede ser útil para seleccionar características relevantes para modelos predictivos y detectar multicolinealidad.

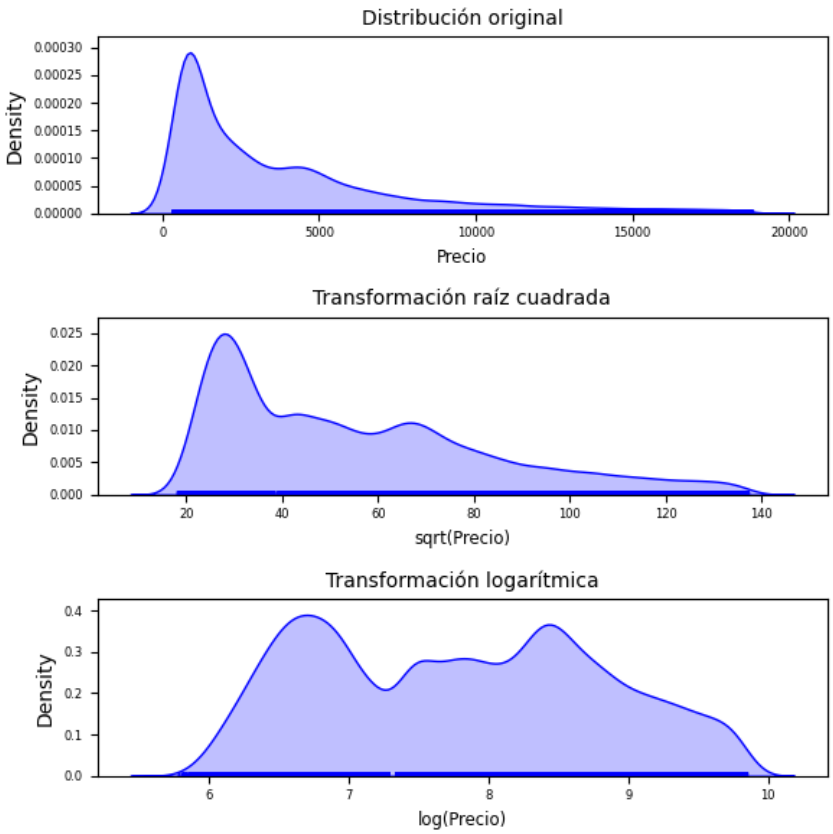

Figura 3: Distribución de la variable objetivo y dos transformaciones de esta (Raíz cuadrada y Logaritmo)

En la figura 4 se puede observar que las variables x, y, z y *carat* tienen fuertes correlaciones entre sí, lo cual puede indicar multicolinealidad. Ahora graficaremos un heatmap para visualizar de formas mas eficiente la correlación entre todos los pares de variables

	variable_1	variable_2	r	abs_r
41	x	y	0.979873	0.979873
49	y	x	0.979873	0.979873
36	x	carat	0.978037	0.978037
4	carat	x	0.978037	0.978037
58	z	x	0.975480	0.975480
42	x	z	0.975480	0.975480
54	z	carat	0.961045	0.961045
6	carat	z	0.961045	0.961045
59	z	y	0.960469	0.960469
51	y	z	0.960469	0.960469

Figura 4: Los 10 pares de variables con mayor correlación en la base de datos

En la figura 5 se pude observar que las variables x, y y z tienen fuertes correlaciones positivas entre sí y con *carat*, lo cual confirma la multicolinealidad

observada anteriormente. *price* tiene una fuerte correlación positiva con *carat*, x, y y z, indicando que estas variables pueden ser buenos predictores del precio. *latitude* y *longitude* no muestran correlaciones significativas con las demás variables.

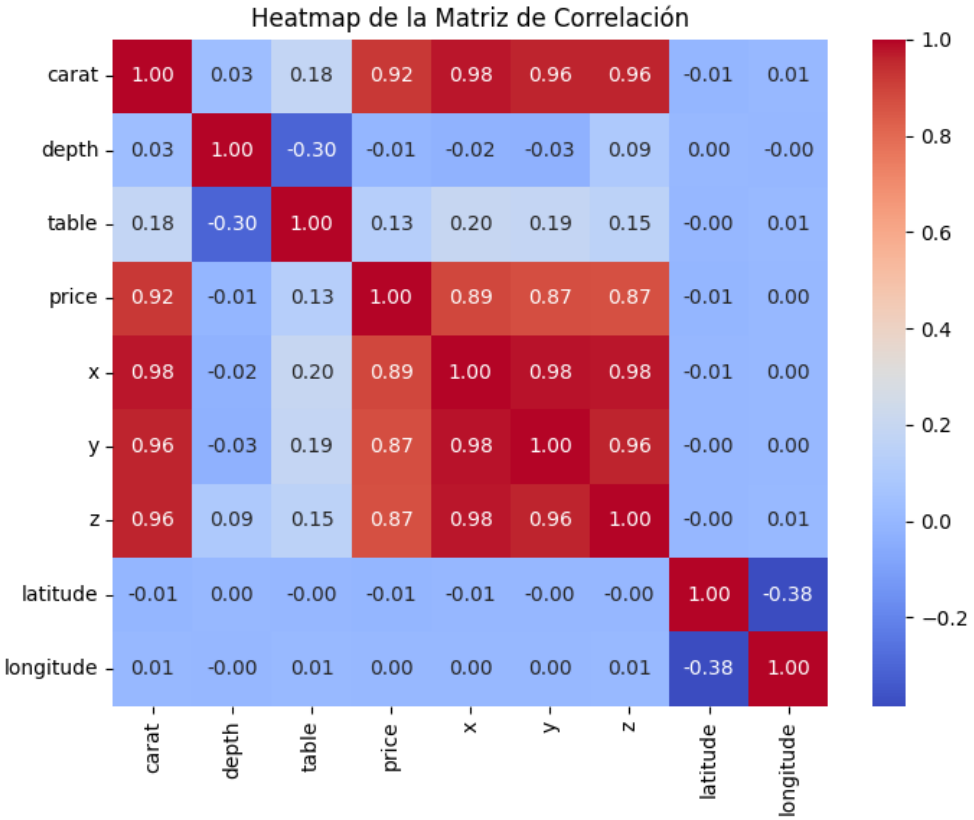

Figura 5: Los 10 pares de variables con mayor correlación en la base de datos

Dado que x, y y z están altamente correlacionadas con *carat* y entre sí, se decide eliminar estas variables del dataset. Esto reduce la multicolinealidad, que puede afectar la interpretación y estabilidad del modelo. Además, usar solo carat para representar el tamaño físico del diamante simplifica el modelo y evita redundancias.

Proceguiremos con el analisis de la distribución de variables cuantitativas. El objetivo es revisar y visualizar la distribución de las variables cuantitativas restantes para entender mejor su comportamiento y determinar si es necesario aplicar transformaciones adicionales al igual que hicimos con la variable objetivo. En la figura 6 se pueden apreciar los siguiente resultados

Distribución de carat: La variable carat muestra una distribución sesgada hacia la derecha con varios picos, lo que sugiere que los datos están agrupados en

ciertas categorías de peso, por lo que se podria decir que tiene una distribución sesgada similar a la variable objetivo price, sugiriendo que podría beneficiarse de una transformación logarítmica.

Distribución de depth: La variable depth muestra una distribución más normal, lo cual es ideal y no requiere transformaciones adicionales.

Distribución de table: La variable table tiene una distribución con picos definidos, lo que indica que los datos están agrupados en ciertos valores de tamaño de la mesa. Se podria decir entonces que esta muestra una distribución que parece una variable discreta disfrazada de continua, indicando que podría necesitar una transformación o tratamiento especial.

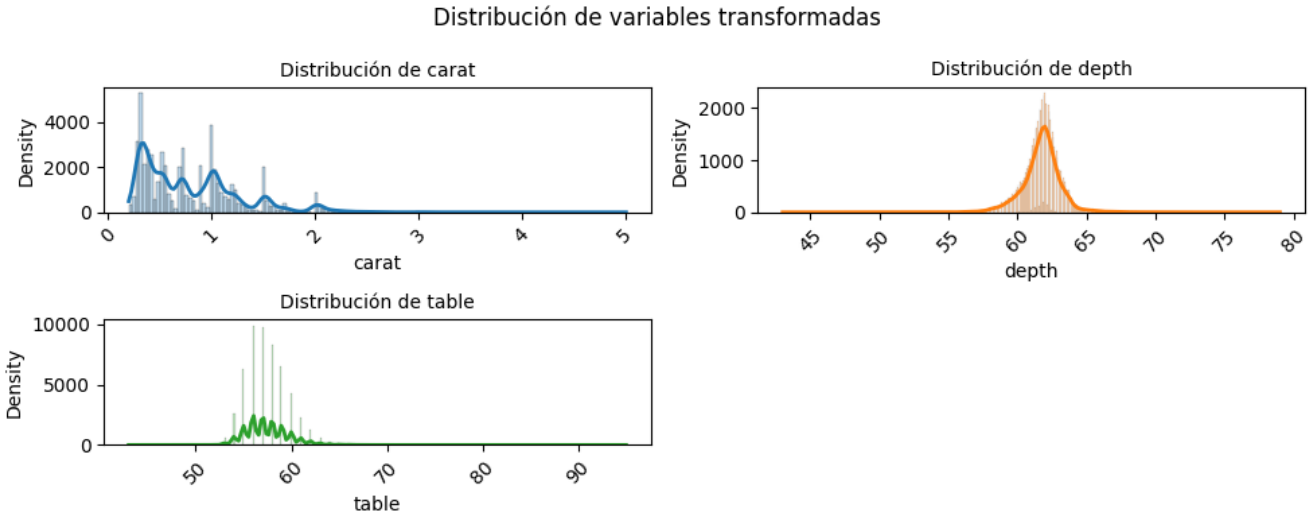

Figura 6: Distribuciones de las variables descriptoras

Ahora realizaremos el Binning de la Variable *table*. El objetivo es simplificar la complejidad de la variable *table* que presenta una distribución con múltiples picos. Para esto, se agrupan los valores de *table* en 5 clusters utilizando el algoritmo de K-means, de manera que se puedan capturar los patrones principales sin perder información significativa. En la figura 7 se aprecia la distribucion de la nueva varible cualitativa.

3.2 Análisis descriptivo Variables Cualitativas

Antes de comenzar a analizar las variables categóricas, se plantea una hipótesis: ."El precio de los diamantes puede variar según la geolocalización de sus compras o ventas". Para probar esta hipótesis, se da una primera visualizacion

de los datos de geolocalización en la figura 8 utilizando la biblioteca `folium` en la figura .

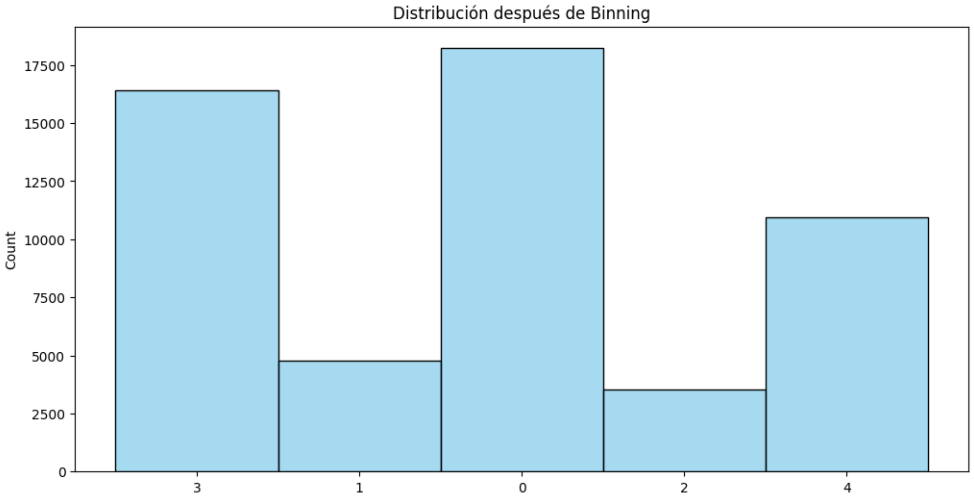

Figura 7: Distribuciones del Binning de la variable *table*

Como se puede apreciar las localizaciones de las ventas o compras de diamantes están presentes por todo el mundo, así que esta hipótesis podrá ser probada filtrando los precios y colocándolos en el mapa para ver cuanto afecta la localización. Sin embargo hay presentes varios datos un poco extraños, como por ejemplo una localización en el centro de la Antártica,lo que indica que se debe proceder con cuidado.

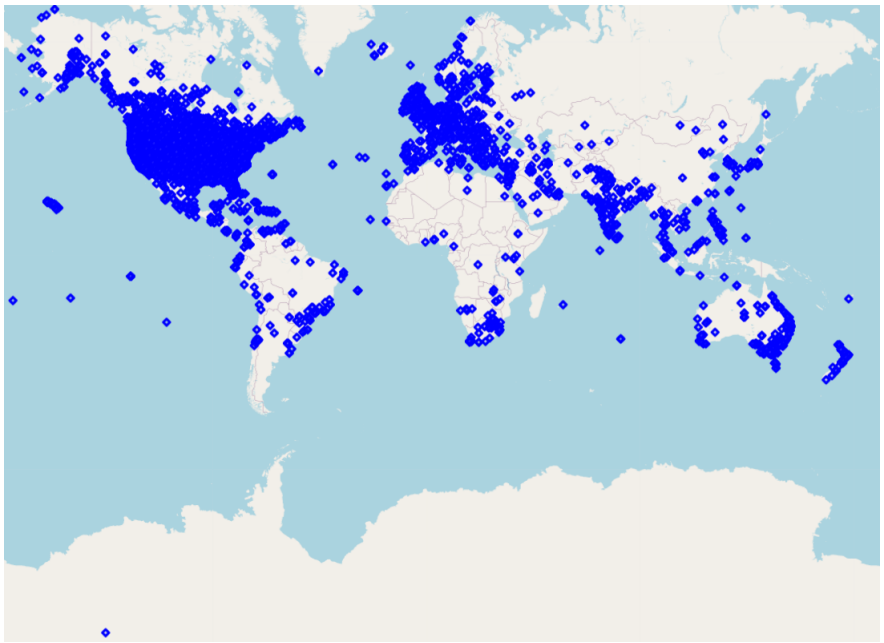

Figura 8: Geolocalización de los robos de diamantes

Primero veremos una gráfica de barra para obtener una imagen de cuan distinto son los precios variando el continente y luego crearemos un mapa para ver la distribución de precios, pues por oferta y demanda en donde mas se vendan puede que haya mas probabilidad hay de que el precio aumente y viceversa.

Para observar si se puede añadir esta información al modelo realizaremos Clasificación de Coordenadas Geográficas por Continentes. El objetivo es clasificar los datos geográficos de los diamantes según los continentes para facilitar el análisis de cómo la ubicación geográfica puede influir en el precio y las ventas de los diamantes.

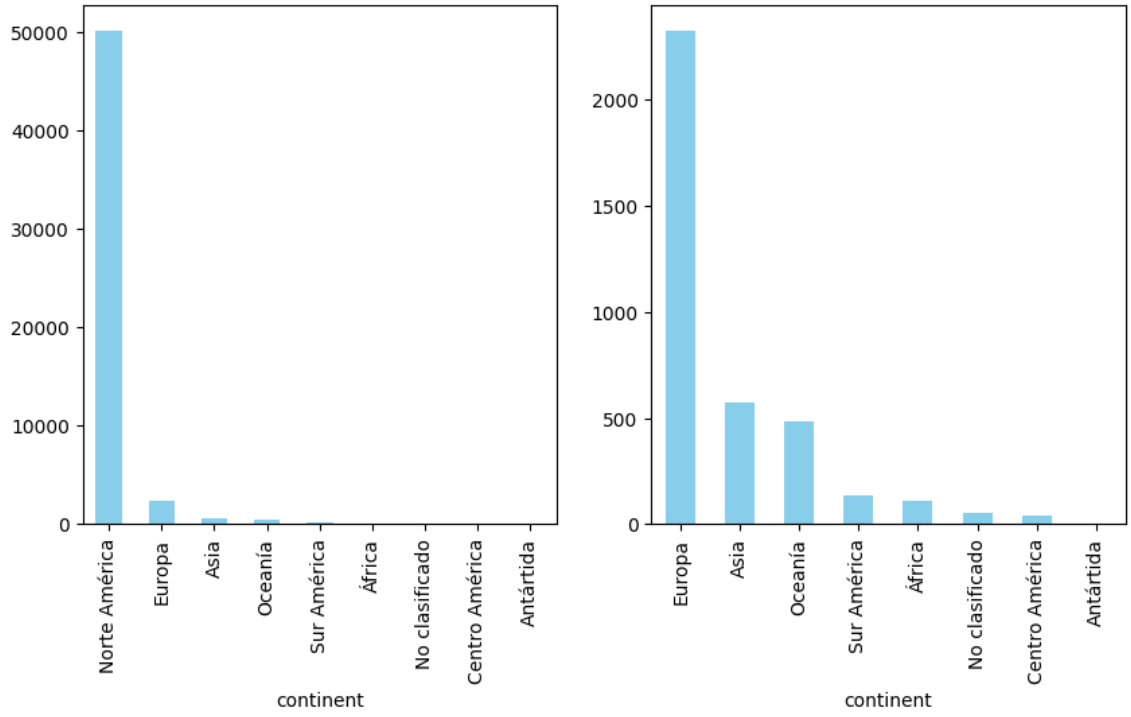

Figura 9: Distribucion de la clasificacion de los robos de los diamantes por continentes

Las visualizaciones muestran la distribución de los datos de diamantes por continente: Norte América tiene la mayor cantidad de datos. Europa, Asia y los otros continentes tienen significativamente menos datos. Segunda Gráfica.Europa es el continente con más datos después de Norte América. Asia y Oceanía también tienen una cantidad considerable de datos. Los demás continentes tienen relativamente pocos datos.

Como se observa en la visualización geográfica, las ventas de diamantes están concentradas principalmente en Norte América y Europa. Sin embargo, al analizar los precios, se nota que no varían significativamente entre continentes,

como se puede apreciar en la figura 10.

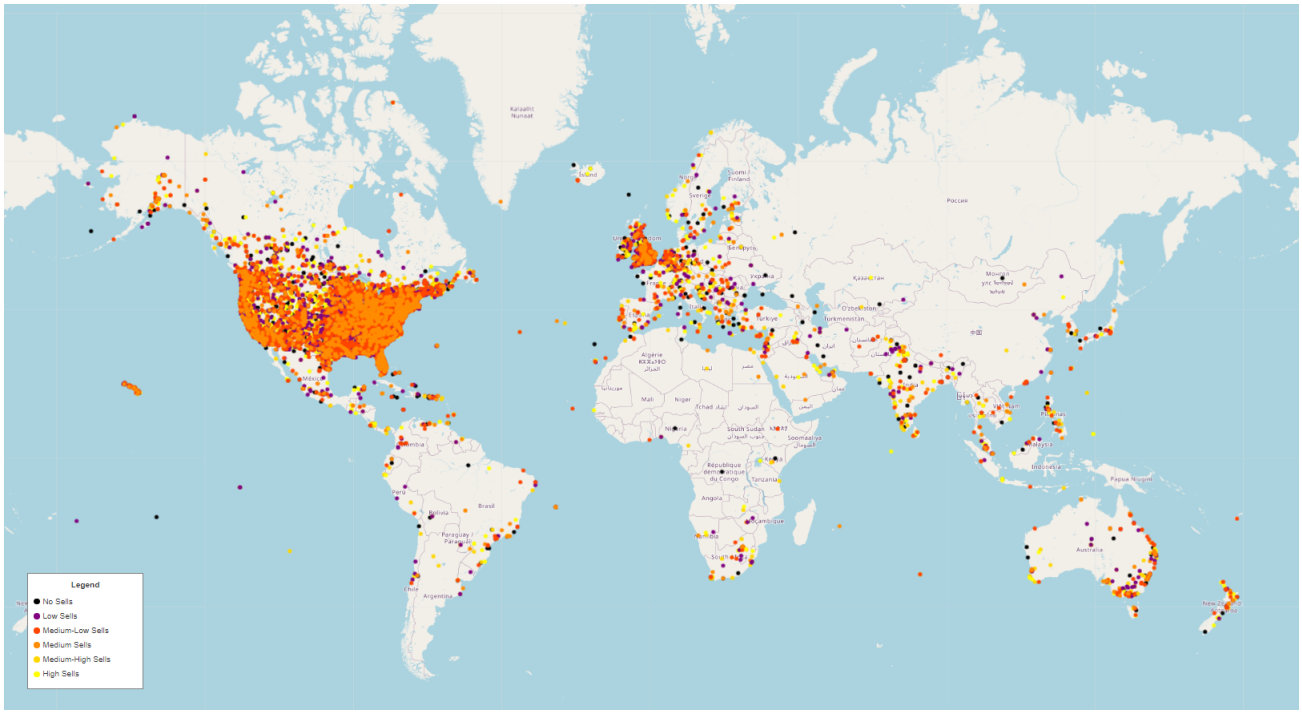

Figura 10: Distribucion geografica del valor de los diamantes

Además, algunas coordenadas parecen estar mal geolocalizadas, lo cual podría introducir ruido en el modelo. Dado que la variable continent está muy desbalanceada y podría no aportar información valiosa para la predicción de precios, se decide eliminar las variables geográficas del análisis para simplificar el modelo. Esta decisión busca simplificar el modelo y mejorar su rendimiento al enfocarse en variables más predictivas y menos ruidosas. Para otros análisis, como estudios de mercado específicos por región, estas variables pueden ser útiles, pero no para el objetivo inmediato de predicción de precios.

El siguiente bloque de código se centra en el análisis descriptivo de variables categóricas en el conjunto de datos. El objetivo es explorar la distribución de las variables *cut*, *color*, y *clarity* para identificar patrones y posibles inconsistencias.

Para cut: La alta frecuencia de Ideal sugiere que los diamantes con el mejor corte son los más comunes en este conjunto de datos, lo que podría indicar una preferencia del mercado o una calidad predominante en los datos. Para color: La predominancia de los colores G, E, y F podría sugerir una preferencia por estos grados de color específicos, posiblemente debido a su equilibrio entre calidad y costo.

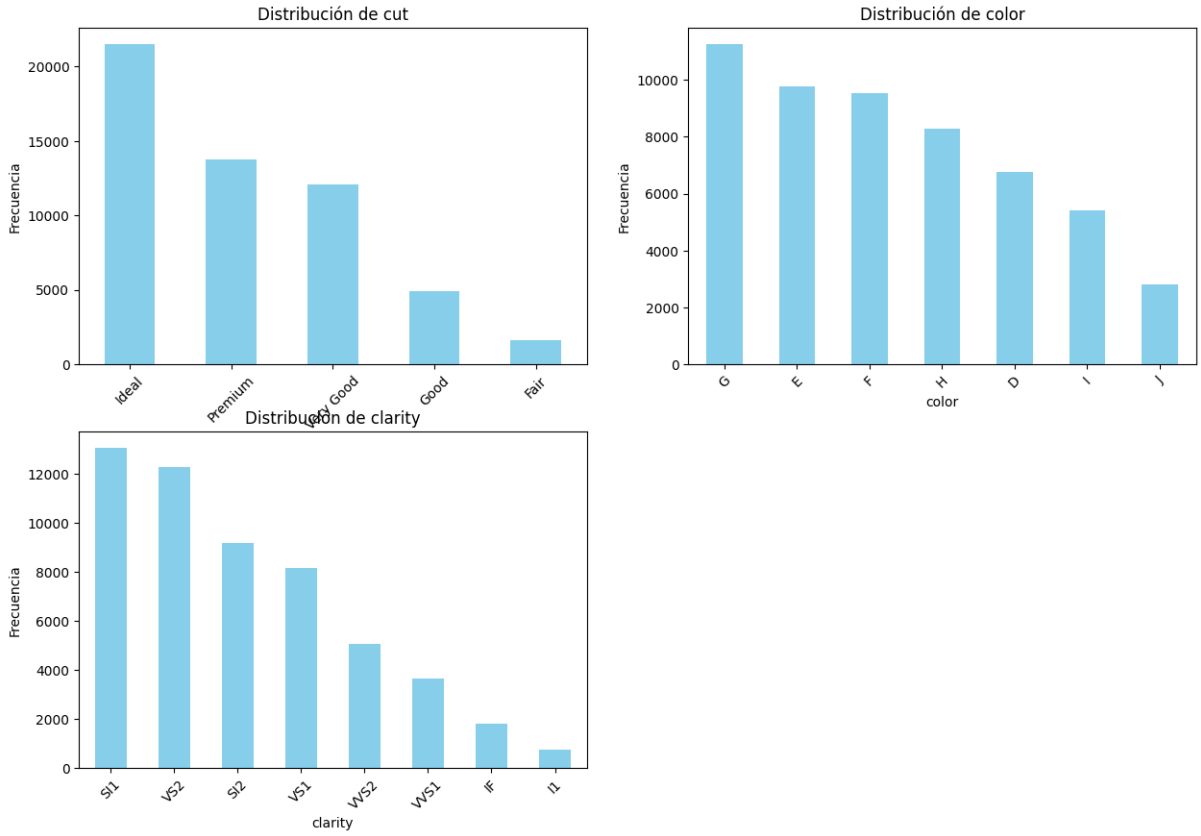

Figura 11: Distribución de variables categóricas

Para clarity: La alta frecuencia de SI1 y VS2 indica que los niveles de claridad con inclusiones ligeras son los más comunes, lo que puede ser una indicación de lo que el mercado considera aceptable en términos de inclusiones y claridad.

4. Modelado

4.1 Creación del Pipline de Preprocesado

Comenzaremos dividiendo el conjunto de datos limpio en conjunto de entrenamiento y de test. La división del conjunto de datos separa los datos en conjuntos de entrenamiento y prueba, con un 80 % de los datos destinados al entrenamiento (train_size=0.8) y el 20 % restante para pruebas. Las características (features) del conjunto de datos se obtienen excluyendo la columna *price*, mientras que la variable objetivo es el precio de los diamantes. Se utiliza random_state=1234 para garantizar la reproducibilidad de los resultados y shuffle=True para barajar los datos antes de la división. Las dimensiones resul-

tantes de los conjuntos son: X_train con 43115 filas y 6 columnas, X_test con 10779 filas y 6 columnas, y_train con 43115 valores, y y_test con 10779 valores, lo que refleja la distribución adecuada de los datos para el entrenamiento y la evaluación del modelo.

Luego se crea un pipeline de preprocesamiento de datos sklearn.pipeline.Pipeline y sklearn.compose.ColumnTransformer. Este pipeline asegura que todas las transformaciones necesarias anteriormente descritas junto con las normalizaciones y los one hots se apliquen a las columnas adecuadas del conjunto de datos de manera eficiente y reproducible para que los ingienieros de datos o los ML Engineer en un futuro puedan implementar un piplines de trabajo de forma sencilla.

4.2 Modelos a usar

En esta sección, se seleccionan modelos relativamente ligeros para poder realizar tuning de hiperparámetros de los mismos. Se utilizan métodos de validación como la validación cruzada simple (vanilla cross-validation) para evitar tiempos prolongados de entrenamiento. La optimización de los hiperparámetros se lleva a cabo utilizando la Optimización Bayesiana con la biblioteca scikit-optimize.

La Optimización Bayesiana construye un modelo probabilístico en el que la función objetivo es la métrica de validación del modelo (como RMSE, AUC, precisión, etc.). Esta técnica dirige la búsqueda hacia las regiones más prometedoras del espacio de hiperparámetros en cada iteración. La optimización de los hiperparámetros se realiza utilizando la biblioteca skopt.

4.2.1 Modelo de Regresión Spline

Supongamos que tenemos una variable predictora x. La transformación spline de x con k nudos puede ser representada como:

$$S(x) = \beta_0 + \sum_{i=1}^{k} \beta_i B_i(x)$$

donde $B_i(x)$ son las funciones base B -spline y β_i son los coeficientes que deben ser estimados.

Después de la transformación de las variables mediante splines, se ajusta un modelo de regresión lineal sobre las variables transformadas.

Si consideramos que $S_j(x)$ es la transformación spline de la j-ésima variable predictora, el modelo lineal puede ser expresado como:

$$y = \alpha + \sum_{j=1}^{p} \sum_{i=1}^{k_j} \beta_{ij} B_{ij}\left(x_j\right) + \epsilon$$

Este modelo permite estimar relaciones no lineales entre las variables mediante el ajuste de polinomios para cada una de las variables predictoras

4.2.2 Modelo XGBoost

XGBoost es una implementación avanzada del algoritmo de boosting de gradiente, que combina múltiples árboles de decisión para mejorar la precisión del modelo. El objetivo es minimizar una función de pérdida, generalmente la cuadrática, agregando árboles secuencialmente. La fórmula general para el modelo de predicción es:

$$\hat{y}_i = \sum_{k=1}^{K} f_k\left(x_i\right), \quad f_k \in \mathcal{F}$$

donde $\mathcal{F}$ es el espacio de árboles de decisión, K es el número de árboles, y f_k es un árbol de decisión.

4.2.3 Modelo Random Forest

Random Forest: Es un conjunto de árboles de decisión entrenados con diferentes subconjuntos del conjunto de datos. La predicción final se obtiene promediando las predicciones de los árboles individuales. La fórmula general del modelo de predicción es:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} h_t(x)$$

donde T es el número de árboles en el bosque y $h_t(x)$ es la predicción del árbol t.

4.2.4 Multi-Layer Perceptron (MLP)

Un Multi-Layer Perceptron (MLP) es un tipo de red neuronal artificial que consiste en una capa de entrada, una o más capas ocultas, y una capa de salida. La arquitectura básica de un MLP incluye una capa de entrada, capas ocultas y capa de salida:.

La capa de entrada recibe los valores de las características del dataset. Supongamos que tenemos p características de entrada $x_1, x_2, \ldots, x_p$

Cada capa oculta está compuesta por múltiples neuronas, y cada neurona aplica una transformación no lineal a la suma ponderada de las entradas.Si la capa oculta tiene H neuronas, las salidas de la capa oculta se pueden representar como:

$$h_j = f\left(\sum_{i=1}^{p} w_{ij}x_i + b_j\right) \quad con \quad j = 1, 2, \ldots, H$$

donde: w_{ij} son los pesos sinápticos. b_j son los sesgos. f es una función de activación (por ejemplo, ReLU, sigmoid).

La capa de salida toma las salidas de la última capa oculta y las transforma en la predicción final. Si el modelo tiene o neuronas de salida (en caso de regresión generalmente $o = 1$ la salida del MLP puede ser representada como:

$$y = g\left(\sum_{j=1}^{H} v_j h_j + c\right)$$

donde: v_j son los pesos de la capa de salida. c es el sesgo de la capa de salida. g es la función de activación de la capa de salida (para regresión, usualmente la identidad $g(x) = x$).

4.2.5 Stacking

El Stacking (apilamiento) es una técnica de ensamblaje de modelos que combina múltiples modelos base (también llamados nivel 0) y un modelo meta (también llamado nivel 1) para mejorar la capacidad predictiva. La idea principal es que el modelo meta aprende a combinar las predicciones de los modelos base de manera óptima.

Si $f_1, f_2, \ldots, f_M$ son los modelos base yg es el modelo meta, el stacking se puede formular como:

1. Predicciones de los Modelos Base:

$$\hat{y}_m = f_m(\mathbf{X}), \quad m = 1, 2, \ldots, M$$

2. Conjunto de Entrenamiento para el Modelo Meta:

$$\hat{\mathbf{Y}}_{\text{train}} = \begin{pmatrix} \hat{y}_1^{(1)} & \hat{y}_2^{(1)} & \cdots & \hat{y}_M^{(1)} \\ \hat{y}_1^{(2)} & \hat{y}_2^{(2)} & \cdots & \hat{y}_M^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{y}_1^{(n)} & \hat{y}_2^{(n)} & \cdots & \hat{y}_M^{(n)} \end{pmatrix}$$

donde $\hat{y}_m^{(i)}$ es la predicción del m-ésimo modelo base para el i-ésimo ejemplo.

3. Predicción Final del Modelo Meta:

$$\hat{y}_{\text{meta}} = g\left(\hat{\mathbf{Y}}_{\text{test}}\right)$$

El modelo meta aprende a combinar las predicciones de los modelos base de tal manera que minimiza el error en el conjunto de validación, proporcionando una predicción final que, en teoría, es mejor que cualquier modelo base individual.

4.2 Entrenamiento

Para el entrenamiento solo se mostrara una tabla con los resultados tanto de la evaluacion como del test de cada uno de los modelos. En el notebook hay mas informacion sobre el proceso de entrenamiento

	Cross Val Metrics	**Test Metrics**		
Modelo	**RMSE**	**RMSE**	**testmetric name**	**testmetric name**
Regresión Spline	1059.99	1069.58	706.06	0.92
XGBoost	574.20	547.61	281.45	0.98
Random Forest	683.33	666.40	0.97	0.97
MLP	696.87	664.28	351.92	0.97
Stacking	-	529.75	274.68	0.98

Tabla 1: Comparación de Modelos

La figura 12 al igual que la tabla 1 muestra una comparación del rendimiento de varios modelos de aprendizaje automático en términos de dos métricas de

error: RMSE (Root Mean Square Error) y MAE (Mean Absolute Error). Los modelos comparados son Stacking, XGBoost, MLP (Multi-Layer Perceptron), Random Forest, y Splines Regression.

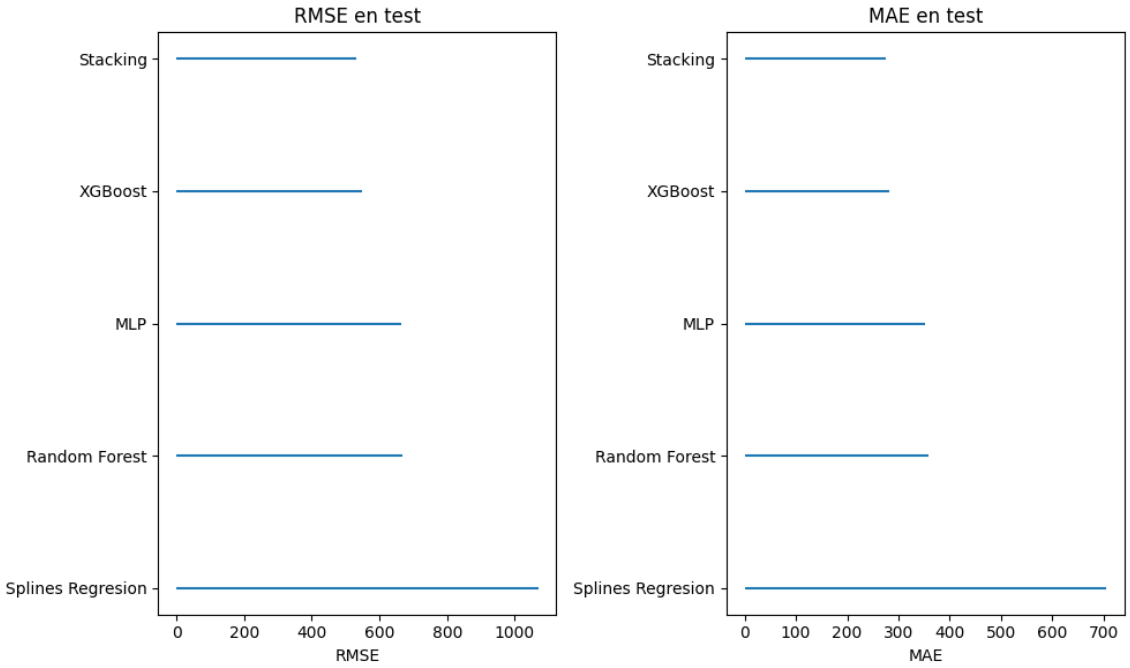

Figura 12: Comparacion de modelos

	Predicción	Real
0	7221.202637	7938.0
1	698.483887	674.0
2	3897.884521	3054.0
3	1241.309570	1397.0
4	533.000244	530.0
...	...	...
10774	950.267822	1040.0
10775	893.642090	781.0
10776	738.835205	576.0
10777	1976.158569	1757.0
10778	4171.383789	4284.0

Figura 13: Distribución de variables categóricas

El modelo de Stacking sobresale en ambas métricas (RMSE y MAE), mostrando los menores errores, lo que sugiere que es el modelo con mejor rendimien-

to entre los evaluados. Splines Regression tiene los mayores errores, indicando que es el modelo con el peor desempeño en este conjunto de datos. XGBoost y Random Forest también muestran buenos resultados, con errores relativamente bajos en comparación con MLP y Splines Regression.

Y por ultimo dando una mirada a una comparacion palpable de los resultados de las predicciones junto con los precios reales, en la figura 13 se puede ver que las salidas son coherentes y cercanas

5. Conclusiones

En este trabajo, se realizaron un exhaustivo proceso de limpieza y preprocesamiento de datos, y un análisis exploratorio que reveló patrones importantes y justificó la eliminación de variables con baja relevancia predictiva. Se entrenaron varios modelos de regresión, incluyendo regresión lineal con splines, Random Forest, XGBoost, MLP y un modelo de Stacking, utilizando optimización bayesiana para el ajuste de hiperparámetros. La comparación de los modelos basada en métricas como RMSE y MAE mostró que el modelo de Stacking tuvo el mejor rendimiento general, seguido de XGBoost y Random Forest. Estos resultados sugieren que los modelos ensamblados, como el Stacking, pueden ofrecer mejoras significativas en el rendimiento predictivo en problemas complejos como la predicción del precio de diamantes. Futuras investigaciones podrían enfocarse en la incorporación de más variables y en la experimentación con otros algoritmos avanzados para seguir mejorando la precisión y la generalización del modelo.