Proyecto Mercado inmobiliario Ruso de Sberbank

Integrantes:

Jhon Jader Caro Sanchez
CC: 1001137636
Ingeniería de Sistemas

Leider Steven Caro Mejía
CC: 1017260248
Ingeniería Civil

José Manuel Ladino Villa
CC: 1010075481
Ingeniería Civil

Inteligencia Artificial

Universidad de Antioquia

2023-2

1. Introducción

1.1 Descripción del problema

En este trabajo el objetivo es predecir el precio de las casas en diferentes zonas de Rusia, esto es especialmente importante en el contexto de una economía rusa volátil y un mercado inmobiliario con características complejas que hacen que las predicciones de precios sean un desafío único. Dicha base de datos fue proporcionada por uno de los bancos más grandes y antiguos de Rusia con el fin de poder hallar modelos que fueran más precisos a la hora de determinar el precio de inmobiliarios y ofrecer dicho servicio predictivo a sus clientes.

1.2 Descripción de la base de datos

Descripción de la base de datos
Usaremos la base de datos de Kaggle de la siguiente competición
https://www.kaggle.com/competitions/sberbank-russian-housing-market que tiene 30472 número de muestras (casas) solo en la base de datos de entrenamiento y 422 columnas

1.3 Variables relevantes de la base de datos

Las variables se dividen en dos bases de datos:
A) ***Variables relacionadas directamente con el inmueble:*** precio de venta (esta es la variable objetivo), identificación de la transacción o venta del inmueble, fecha de la transacción, área total en metros cuadrados, incluidas logias, balcones y otras áreas no residenciales, superficie habitable en metros cuadrados, excluyendo logias, balcones y otras áreas no
residenciales, piso: para apartamentos, piso del edificio, número de pisos del edificio, año de construcción, número de salas de estar, área de cocina, condición del apartamento, nombre del distrito, entre otras .

B) ***Variables relacionadas con la economía al momento de la venta del inmueble:*** Fecha de la transacción, Precio del Petróleo crudo (Rublo), Crecimiento del PIB real e Indicador de inflación, entre otras.

1.4 Métrica de validación

Como métrica de machine learning, se utilizará el RMSLE (Root Mean Squared Logarithmic Error), que es la métrica definida por la competencia en cuestión. El RMSLE mide el error promedio de los logaritmos de las predicciones y los valores reales. Esto tiende a penalizar de manera más significativa las grandes discrepancias en valores más grandes.

1.5 Criterio deseable de desempeño en producción

En general, se espera que el modelo tenga un error de predicción bajo, inicialmente
esperamos que el RMSLE este por debajo del 0.15. Esto significa que las predicciones del modelo deben estar lo más cerca posible de los valores reales de los precios de las casas.

1.6 Resumen ejecutivo

Este proyecto tiene como objetivo principal desarrollar un sistema de predicción de precios de viviendas, utilizando datos relacionados con las propiedades, el entorno y factores económicos en el momento del registro del precio del inmueble. Comenzamos con una exploración exhaustiva de los datos disponibles, identificando patrones y relaciones clave. Luego, realizamos una limpieza detallada para abordar cualquier inconsistencia o valor atípico que pudiera afectar la precisión de los modelos predictivos.

Para lograr este objetivo, hemos implementado varios modelos de machine learning, como vecinos más cercanos (KNN), Random Forest, XGBoost y una red neuronal con PCA para reducir la dimensionalidad de

los datos. Estos modelos se han evaluado minuciosamente en términos de precisión y capacidad para generalizar patrones en la predicción de precios de viviendas.

Nuestra meta es identificar el modelo más efectivo para predecir con precisión los precios de las viviendas en base a las características seleccionadas. Esta metodología estructurada nos permite determinar el modelo que ofrece estimaciones más fiables y valiosas en el mercado inmobiliario.

2. Exploración descriptiva del dataset

2.1 Exploración inicial

Para este primer paso se hará una exploración superficial de la primera base de datos la cual esta nombrada como *Train.csv* la cual tiene variables que son directamente relacionadas con las caracteristicas del inmueble. Lo primero que se realizó como análisis exploratorio fue visualizar las primeras 5 filas, luego visualizamos las columnas, el tipo de dato y la cantidad de valores no nulos, la cantidad de columnas y la cantidad de filas. Se observó que hay 30471 registros y existen 292 variables entre las que se encuentra la variable objetivo la cual se llama *price_doc.*

Luego se realizó el mismo procedimiento con la segunda base de datos que se llama *macro.csv,* esta base de datos contiene la información de los datos economicos de Rusia desde Enero del 2010 hasta Octubre del 2016, contenia variables como el precio del combustible, el crecimiento economico de Rusia, indicadores de PIB, precio del Rublo con respecto al Dolar Estadounidense y entre otras caracteristicas. Se visualizó la cantidad de datos no nulos de cada columna, el tipo de variable, cantidad de columnas y cantidad de filas. Se determinó que la base de datos contiene 100 columnas y 2484 registros.

2.2 Análisis descriptivo

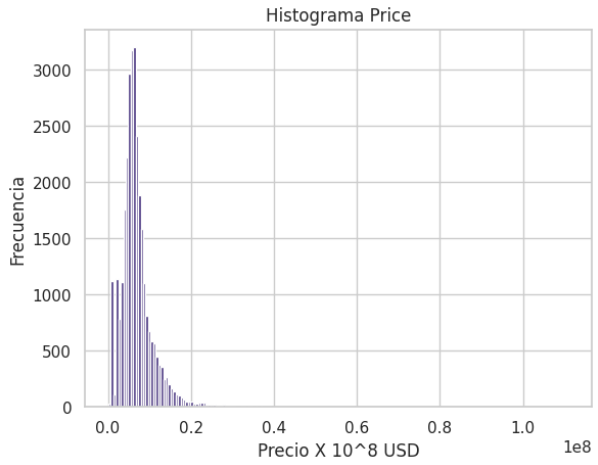
se llevó a cabo un análisis descriptivo de las variables numéricas en el conjunto de datos consolidado. Este análisis abarcó la obtención del conteo total de observaciones, el cálculo de la media y la desviación estándar, la identificación de los valores mínimo y máximo, así como la determinación de los percentiles 25, 50 (mediana) y 75.
Estos aspectos proporcionaron una comprensión profunda de la distribución y la variabilidad de los datos, lo que resultó esencial para orientar las decisiones en etapas posteriores del proyecto.

	full_sq	life_sq	floor	max_floor	material	build_year	num_room	kitch_sq	state
count	30471.000000	24088.000000	30304.000000	20899.000000	20899.000000	1.686600e+04	20899.000000	20899.000000	16912.000000
mean	54.214269	34.403271	7.670803	12.558974	1.827121	3.068057e+03	1.909804	6.399301	2.107025
std	38.031487	52.285733	5.319989	6.756550	1.481154	1.543878e+05	0.851805	28.265979	0.880148
min	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000e+00	0.000000	0.000000	1.000000
25%	38.000000	20.000000	3.000000	9.000000	1.000000	1.967000e+03	1.000000	1.000000	1.000000
50%	49.000000	30.000000	6.500000	12.000000	1.000000	1.979000e+03	2.000000	6.000000	2.000000
75%	63.000000	43.000000	11.000000	17.000000	2.000000	2.005000e+03	2.000000	9.000000	3.000000
max	5326.000000	7478.000000	77.000000	117.000000	6.000000	2.005201e+07	19.000000	2014.000000	33.000000

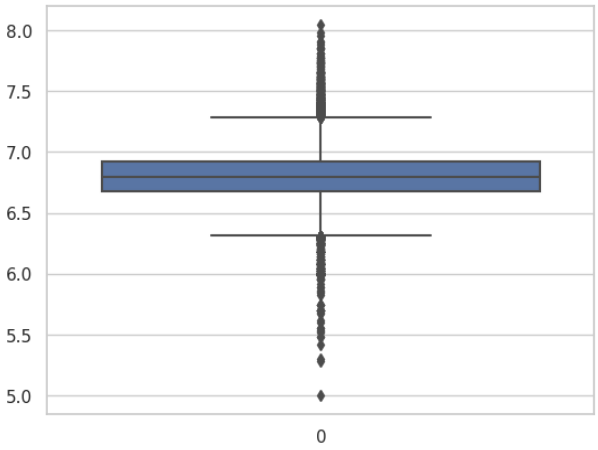
Se identificó inicialmente una cantidad de datos nulos que ascendía a 504,537 observaciones, lo que representaba aproximadamente el 4.2% del total de datos disponibles. No obstante, para cumplir con el objetivo establecido, era necesario incrementar deliberadamente la proporción de datos nulos hasta alcanzar el 5%.

2.3 Histograma de la variable respuesta

Se realizó un un histograma para conocer la distribución de los datos de la variable respuesta se obtuvo el siguiente gráfico



En nuestro caso vemos que la mayoria de los datos se ubican entre $100.000 y $20'000.000 USD aproximadamente.

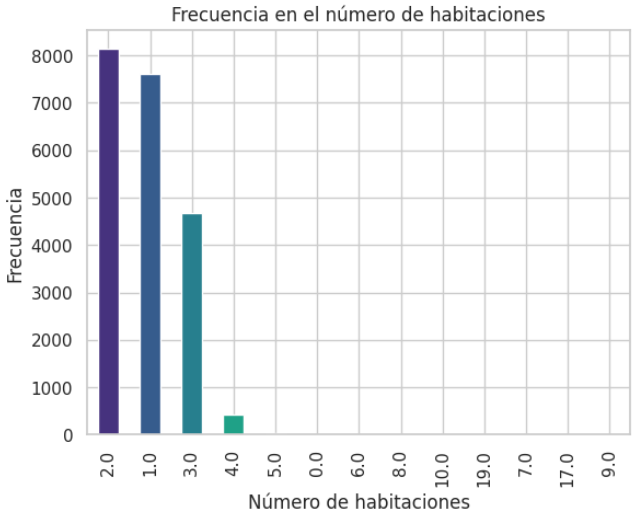


También se elaboró con la misma variable respuesta un gráfico de densidad y un gráfico de caja se determino que existe una cantidad significa de valores atípicos en nuestras muestras.

2.4. Análisis de variables categóricas

En nuestro caso se usarón las variables con menos de 20 categorias y aquellas que fueran de tipo string, para este análisis visual se usarón gráficas de barra para representar la frecuencia de cada catégoria.
Se analizó la variable material de inmueble y se encontró que el 50% de los inmuebles estaban fabricadas del material tipo 1.
Del análisis de la variable estado del inmueble se pudó concluir que la tendecia de los inmuebles se encuentra entre mal estado, regular estado y buen estado, siendo excelente estado la catégoria con menos recurrencias.

Luego de la variable número de habitaciones se puede concluir que las casas con 2 habitaciones son las más recurrentes abarcan un poco más del 28% de las muestras, las casas con una habitación representan el 25% de las muestras, las casas con 3 habitaciones representan el 15% de las muestras y por último las casas con 4 habitaciones representan aproximadamente el 2% de las muestras.



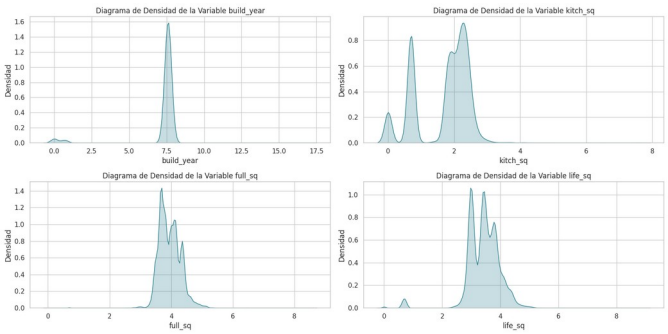
Luego de la variable tipo de producto se puede concluir que casi el 2/3 de las muestras son de tipo inversión mientras el 1/3 de las muestras son de tipo dueños propios de la residencia.

2.5.Análisis de variables númericas

El siguiente procedimiento que se ejecutó fue representar algunas variables númericas con diagramas de densidad para conocer un poco su distribución y saber si algunas de estas variables tenían sesgo o una distribución atipica.

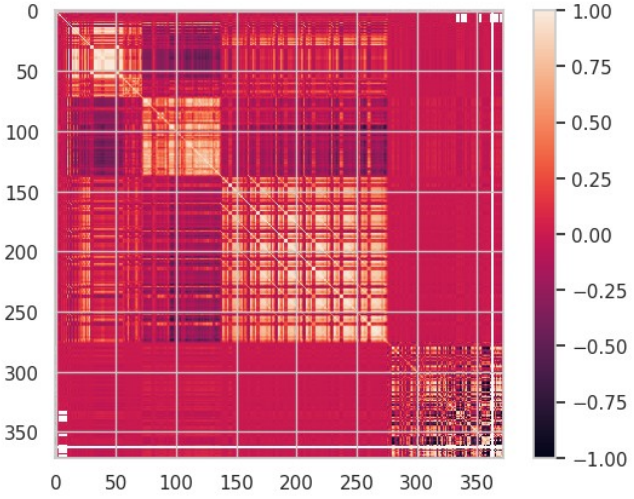
Algunas variables como año de construcción, metros cuadrados de la cocina, metros cuadrados totales y metros cuadrados habitables poseían un sesgo hacia la

izquierda y por ende se decidio aplicar la función **Log(x+1),** y el resultado fue el siguiente:



2.6.Matriz de correlación general

Luego se procedió a contruir una matriz de correlación y se represento gráficamente en un mapa de calor y este fue el resultado:

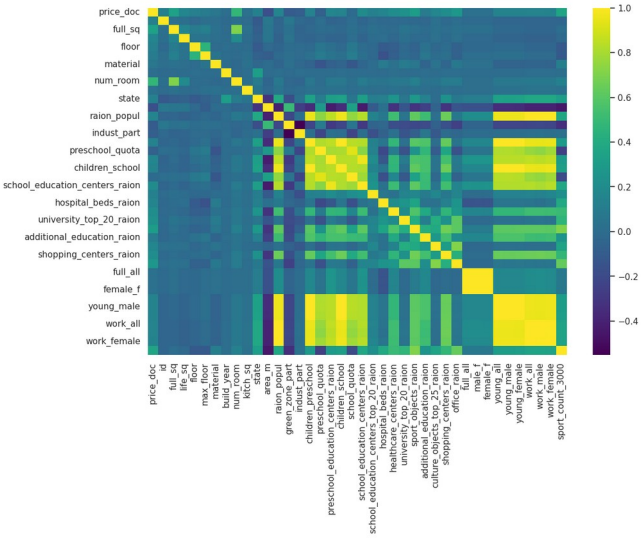


Vemos que el mapa de calor estaba segmentado por zonas, zonas en las que existió una correlación bastante fuerte con otras variables y zonas donde la correlación negativa es bastante pronunciada, y zonas donde no existió correlación entre variables.

2.7.Matriz de correlación segmentada

Luego se realizó un mapa de calor basandonos en la segmentación del anterior mapa de calor general y se tomo el segmento (0,0) para poderlo gráficar y analizarlo más a detalle

En la primera fila vemos a la variable respuesta, vemos que no existe mucha relación de ninguna de estas variables con esta variable respuesta, lo cual significa que no muchas variables de estan pueden explicar el comportamiento del precio de los inmuebles. Tambien se pudó evidenciar que muchas variables independientes tienen correlación muy alta entre ellas.



Luego se tomó el ranking de variable dentro de este mismo segmento con mayor correlación entre variables independientes, se evidenció que existian caracteristicas con más del 95% de correlación.

3. Iteraciones de desarrollo

3.1 Preprocesado de datos

3.1.1 Datos faltantes iniciales

En el dataset principal, denominado *Train.csv*, se observaron ciertas columnas con un número significativo de datos faltantes. Específicamente, las columnas que presentaron la mayor cantidad de valores ausentes fueron el *estado del inmueble* con 13,559 datos faltantes, la *cantidad de camas de hospital en el distrito* con 14,441 datos faltantes y el *año de construcción* con 13,605 datos faltantes. En conjunto, estos valores nulos representaron un total de 261,026, lo que corresponde al 2.93% de los datos en

este dataset. Estos hallazgos subrayan la importancia de abordar la imputación de datos en estas columnas para garantizar que el dataset esté completo y sea adecuado para el análisis y la construcción del modelo predictivo.

En relación al dataset secundario, se identificaron columnas con un notable número de datos faltantes. Específicamente, las columnas con la mayor cantidad de valores ausentes fueron *grp_growth* con 1,023 datos faltantes, *salary_growth* con 658 datos faltantes y *pop_migration* con 658 datos faltantes. En conjunto, estos valores nulos sumaron un total de 46,658, lo que representa aproximadamente el 18.78% de los datos contenidos en este dataset. Esta alta proporción de datos faltantes en "macro.csv" resalta la necesidad de abordar la imputación de estos valores de manera efectiva para asegurar la integridad de los datos en el proceso de análisis y modelado subsiguiente.

3.1.2 Corrección de datos inexactos

En el dataset original, se identificaron valores erróneos relacionados con la ciudad de Tverskoe. Cuando un registro contenía esta designación, los valores de distancia asociados eran inexactos, lo que comprometía la precisión de los datos. Para abordar esta problemática, se ha optado por realizar una actualización utilizando el archivo '*BAD_ADDRESS_FIX.xlsx*', el cual fue obtenido a través del foro de discusión de la competición. Este archivo de corrección proporciona información valiosa para rectificar los datos incoherentes relacionados con la ciudad de Tverskoe y, así, mejorar la calidad y confiabilidad del conjunto de datos en cuestión.
Se aplicó el método *update* para efectuar dicha correción

3.1.3 Merge del dataset final

Para combinar y enriquecer los datos, se llevó a cabo un proceso de fusión (merge) de los datasets *macro.csv* y *Train.csv* en función de la fecha. Este proceso permitió la creación de un nuevo dataset consolidado con un total de 392 columnas y 30,471 filas. La fusión se basó en la coincidencia de las fechas en ambos datasets, lo que posibilitó la incorporación de información económica y macroeconómica de "macro.csv" a las observaciones correspondientes en "Train.csv". Esta integración de datos es fundamental para enriquecer el análisis y la construcción de un modelo predictivo sólido que tenga en cuenta tanto las características de las propiedades como el contexto económico en el momento de su venta. El nuevo dataset resultante proporciona una base más completa y rica para el análisis y modelado subsiguiente.

3.1.4 Generación de datos nulos sintéticos

Para este proceso lo que se hizó fue seleccionar las columnas que ya tuvieran datos nulos, luego de tenerlas identificadas se calculó la cantidad de datos que era necesaria eliminar para lograr el 5% de datos nulos, esta cantidad fue 85.214, se dividió esta cantidad por la cantidad de columnas con datos nulos y aleatoriamente, pero aplicando una semilla para que esta misma aleatoriedad sea reproducible, se eliminaron los datos para completar la cantidad de datos faltantes mínima. Este proceso se hizo en el notebook ‘*02_Preprocesamiento.ipynb’*

3.1.5 Descargar la base de datos completada

Se descarga la base de datos fusionada y con la cantidad de valores nulos completada, se descargó la base de datos para poder

continuar el proceso en el segundo cuaderno llamado *02-Preprocesamiento*

3.1.6 Generación de variables categóricas sintéticas

Esta segunda parte esta alojada en el segundo cuaderno llamado *02-Preprocesamiento.*
Otro de los requisitos mínimos es que el 10% de las variables sean categóricas, por ende se analizó que variables son categóricas y cuantas variables categóricas hacían falta para componer el mínimo solicitado. El mínimo solicitado para este caso son 39 columnas categóricas. Para completar el mínimo estipulado se convirtieron 24 columnas a categóricas, formulando la siguiente pregunta ¿el valor es mayor a la media de su columna? En caso de que fuera cierto el valor cambiaría a 1, en caso contrario tomaría el valor de 0.

3.1.7 Manejo de valores nulos

Después de una evaluación minuciosa, se decidió eliminar aquellas variables que contenían más del 58% de datos perdidos. Esta medida se tomó para preservar la calidad general de nuestro conjunto de datos y evitar posibles distorsiones en los resultados.
Para las variables categóricas ordinales con valores nulos, se optó por imputar la mediana de los datos disponibles. Este enfoque se eligió por su capacidad para mantener la coherencia y la tendencia central de esta categoría de variables.

En cuanto a las variables categóricas no ordinales, se procedió a rellenar los valores faltantes con la moda de cada variable. Esta estrategia se implementó para preservar la distribución predominante de los datos categóricos no ordinales.

Las variables numéricas con valores nulos fueron tratadas de acuerdo con su distribución. Aquellas que mostraron una distribución normal fueron rellenadas con el promedio de los valores existentes. Por otro lado, las variables numéricas sin una distribución normal fueron completadas utilizando la mediana, lo que garantizó la preservación de la tendencia central y minimizó el impacto de valores atípicos en el conjunto de datos.

3.1.8 Corrección de valores atípicos

Durante la preparación de los datos, se identificaron irregularidades en variables numéricas, como el uso de "," en lugar de "." para representar decimales y la presencia de valores como "#!". Estos fueron corregidos para asegurar la interpretación correcta de los datos. Se aplicó una transformación para reemplazar "#!" con NaN y se empleó un método de 'backward fill' para completar los valores faltantes en estas variables.

3.1.9 Codificación de variables

El análisis de las variables reveló la presencia de 16 variables tipo 'object' que requerían ser codificadas para su uso efectivo en modelos de inteligencia artificial.
Para la variable 'Nombre del Sector' y 'Tipo de Inversión', se aplicó el método de 'dummy encoding', generando nuevas columnas binarias para cada categoría única. Esta técnica permitió representar estas variables de manera numérica sin asumir una relación ordinal entre las categorías.

Se empleó una estrategia de binarización en 12 variables que indicaban la presencia de servicios específicos en la propiedad. Esto resultó en nuevas variables binarias que reflejaban la presencia o ausencia de cada servicio.

La variable de fecha se desglosó en tres variables numéricas separadas para el día, mes y año. Esta transformación facilitó el análisis temporal sin perder información relevante.

Para la variable que indicaba el nivel de ecología cercano al inmueble, se asignaron valores jerárquicos numéricos para reflejar la escala cualitativa ('no_data', 'pobre', 'satisfactorio', 'bueno', 'excelente'). Esta asignación se realizó en una escala de 0 a 1, donde 'no_data' corresponde a 0 y 'excelente' a 1, estableciendo valores intermedios de manera proporcional (0.25, 0.5, 0.75) para reflejar la gradación entre las categorías.

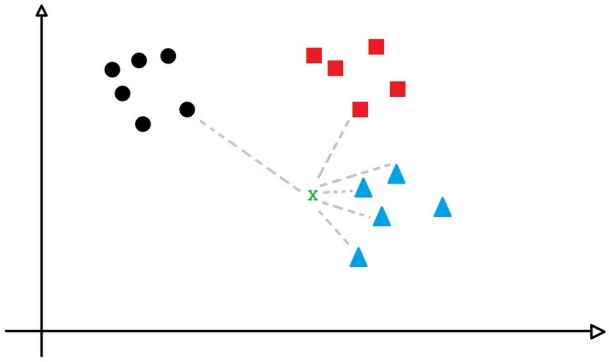
3.2 Modelos supervisados

3.2.1 Modelo KNN (K-Nearest Neighbors) en Predicción de Precios de Casas

El modelo KNN es una herramienta efectiva en la predicción de precios de casas debido a su enfoque intuitivo y la naturaleza del problema. Este algoritmo se basa en la premisa de que objetos similares tienden a estar cercanos en un espacio multidimensional.

En el contexto de precios de casas, la idea es que propiedades con características similares tienden a tener valores similares. KNN busca relaciones entre características como área, ubicación, número de habitaciones, entre otras, y estima el precio basándose en propiedades cercanas en este espacio multidimensional.
KNN no hace suposiciones específicas sobre la distribución de los datos. En el caso de los precios de las viviendas, donde la relación entre características y precio puede ser compleja o no lineal, KNN puede adaptarse bien a esta complejidad al no imponer restricciones sobre la forma de los datos.

Este modelo lo desarrollaremos en el notebook *03-Modelo KNN.ipynb*



3.2.1.1 Modelo KNN inicial

Como primer paso en el modelado, dividimos nuestros datos en dos conjuntos: un 75% para entrenamiento y un 25% para validación. Luego, configuramos un modelo KNN que estimaba utilizando los 5 vecinos más cercanos, una técnica comúnmente aplicada en problemas de regresión.

Utilizamos una métrica de validación específica, la raíz cuadrada de la media cuadrática logarítmica del error (RMSLE), sugerida por la competencia de Kaggle en la que participamos. Este enfoque nos permitió evaluar la precisión de nuestro modelo en relación con el estándar de la competencia.

Sin embargo, los resultados no alcanzaron el nivel de satisfacción deseado, ya que el modelo obtuvo un puntaje de validación de 0.5634. Esto nos indica que, aunque el modelo KNN fue aplicado con una configuración inicial estándar, su capacidad para predecir los precios de las casas no fue óptima según la métrica de evaluación establecida.

3.2.1.2 Optimización del modelo KNN

Para mejorar la efectividad del modelo KNN, implementamos una serie de estrategias. En primer lugar, reconocimos la importancia de estandarizar las variables antes de aplicar el algoritmo KNN.

En el modelo KNN, las variables con diferentes escalas y rangos pueden tener un

impacto desigual en la medida de distancia entre puntos. La estandarización es crucial ya que iguala la importancia relativa de cada característica, al reescalarlas para que tengan una media de cero y una desviación estándar de uno. Esto asegura que ninguna variable domine artificialmente el cálculo de distancias.

Luego de este reconocimiento, volvimos a dividir el conjunto de datos en un 75% para entrenamiento y un 25% para validación. Posteriormente, entrenamos el modelo KNN utilizando las variables estandarizadas.

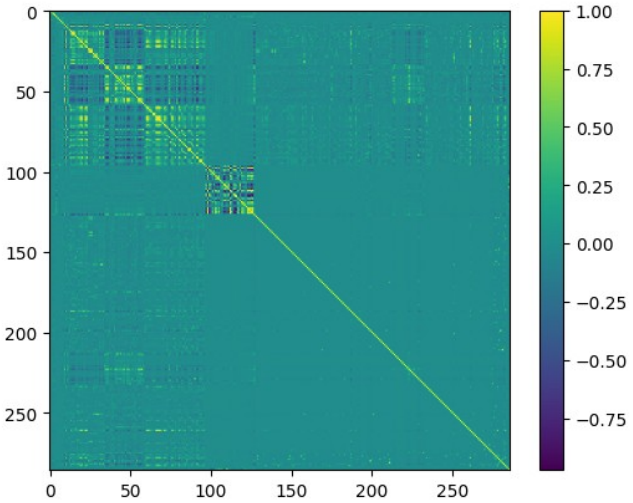
Los resultados fueron alentadores: obtenemos un valor de RMSLE de **0.5463,** lo que indica una mejora notable con respecto a la configuración anterior del modelo. Esta mejora en la métrica de evaluación sugiere que la estandarización de las variables tuvo un impacto positivo en la capacidad predictiva del modelo KNN para estimar los precios de las casas.

3.2.1.3 Eliminación de variables independientes con alta correlación entre ellas

En un proceso de análisis de datos o modelado predictivo, es común eliminar variables independientes que muestran una alta correlación entre sí, ya que esta multicolinealidad puede dificultar la interpretación del modelo y reducir su capacidad predictiva. Para abordar este problema, se realiza una selección de características que implica identificar las relaciones lineales entre las variables independientes y la variable de respuesta.

En este proceso, se analiza la correlación entre las variables independientes y se retiene solo la que exhibe la mejor correlación con la variable de respuesta. Esto se hace para simplificar el modelo y preservar la relevancia de la información más fuertemente relacionada con la variable objetivo.

El resultado de esta selección de características es un conjunto de variables independientes optimizado, lo que puede mejorar la precisión y la interpretabilidad del modelo, al tiempo que reduce la redundancia de la información contenida en las variables independientes. Este enfoque de selección de características es una práctica común en la ciencia de datos y el aprendizaje automático para mejorar la calidad de los modelos predictivos.



Al aplicar esta técnica, observamos una reducción sustancial en el Error Cuadrático Medio de la Raíz Logarítmica (RMSLE) del modelo, logrando una disminución hasta **0.534.** Esta mejora representa una reducción de 0.01 en el RMSLE, lo que indica una mayor precisión en las predicciones de precios de las casas.

Además de la mejora en la precisión, esta estrategia nos brindó beneficios adicionales al reducir la complejidad del modelo. Pasamos de trabajar con un conjunto de datos que contenía un poco más de 500 variables a uno más manejable de 284 variables. Esta reducción en la dimensionalidad no solo optimizó el rendimiento del modelo, sino que también simplificó su interpretación y su capacidad de generalización.

3.2.1.4 Intento de hiperparametrización con GridSearch de KNN

Se llevó a cabo un intento exhaustivo de optimizar el modelo KNN mediante una búsqueda de hiperparámetros utilizando la técnica de GridSearch. Se configuraron múltiples combinaciones de parámetros, incluyendo 'n_neighbors', 'weights', 'algorithm', 'leaf_size', 'metric' y 'p', con el objetivo de encontrar la configuración óptima que maximizara el rendimiento del modelo.
Resultados del Intento de Hiperparametrización

Se ejecutaron iteraciones del modelo con 5 validaciones cruzadas para cada conjunto de parámetros, sin embargo, debido a la complejidad de las combinaciones y la cantidad de modelos a entrenar, el proceso resultó en un alto gasto computacional.

Tras transcurrir un periodo extenso de tiempo (aproximadamente 12 horas) en un entorno de Google Colab, la sesión del entorno virtual se desconectó antes de obtener resultados concretos. Esta interrupción impidió la finalización exitosa del proceso de optimización de hiperparámetros, limitando la posibilidad de obtener conclusiones o mejoras significativas en el modelo KNN.

3.2.1.5 Hiperparametrización con Randomized Search

Tras el intento fallido de hiperparametrización con GridSearch, recurrimos a una técnica más eficiente conocida como Randomized Search. Este enfoque nos permitió explorar un espacio de hiperparámetros de manera más efectiva y, en este caso, arrojó resultados significativos en un tiempo reducido.
Resultados del Randomized Search

Después de apenas 20 minutos de ejecución, logramos identificar los mejores parámetros para nuestro modelo KNN:

n_neighbors	weights	p	algorithm	leaf_size	metric	Tiempo de ejecución	RMLSE
13	distance	2	kd_tree	30	minkowski	0.977624	0.530694
12	distance	2	kd_tree	30	euclidean	0.956752	0.530764
12	distance	1	auto	30	euclidean	0.035069	0.530764
11	distance	1	brute	10	euclidean	0.033249	0.531029
13	distance	1	brute	30	minkowski	0.033603	0.531749

Este modelo optimizado tuvo el tiempo de ejecución más alto, aproximadamente de 0.978 segundos para entrenarse. Sin embargo, esta inversión en tiempo resultó en mejoras sustanciales en la capacidad predictiva del modelo.

Con estos parámetros optimizados, logramos reducir significativamente el RMSLE en 0.004 unidades con respecto a la configuración previa. Hasta el momento, alcanzamos un RMSLE de 0.5307.

3.2.1.6 Eliminación de variables con poca correlación a la variable respuesta del modelo KNN

Para mejorar la capacidad predictiva del modelo, llevamos a cabo una selección de variables basada en su correlación con la variable objetivo 'Price_doc'. Se procedió a eliminar aquellas características que mostraban una correlación menor a 0.25 con 'Price_doc'. Además, consideramos eliminar aquellas variables con correlaciones negativas comprendidas entre 0 y -0.1, ya que, aunque presentaban una relación inversa, esta asociación era mínima.

La eliminación de variables poco correlacionadas con la variable objetivo puede mejorar el modelo por varias razones, Variables con baja correlación a menudo contienen ruido o información insignificante para predecir la variable objetivo. Al eliminarlas, reducimos la interferencia y el potencial de sobreajuste del modelo. También Al conservar solo las variables más correlacionadas, permitimos que el modelo

se centre en las características más relevantes y significativas para la predicción de 'Price_doc'. Esto puede mejorar la precisión de las estimaciones.

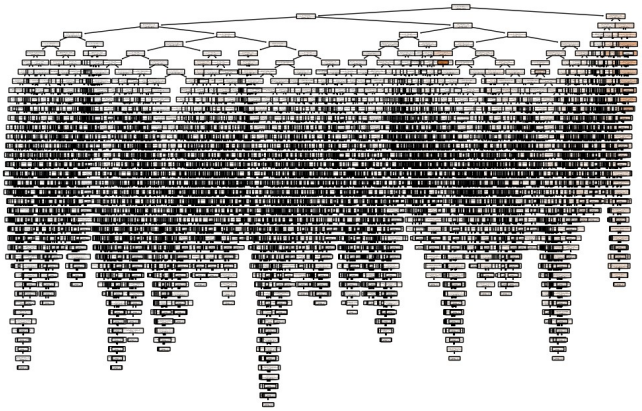
Otro motivo es que al reducir el número de variables, simplificamos la complejidad del modelo, lo que puede resultar en una mejor generalización y evita la redundancia de información entre características poco útiles.

Tras aplicar la estrategia de eliminar variables con baja correlación respecto a la variable objetivo 'Price_doc', logramos reducir significativamente la dimensionalidad del conjunto de datos. De un conjunto inicial de variables, nos quedamos con un conjunto más concentrado de 27 características relevantes y altamente correlacionadas con 'Price_doc'.

Esta reducción en la cantidad de variables tuvo un efecto positivo en el rendimiento del modelo. Experimentamos una notable mejora en el RMSLE, reduciéndolo en 0.0382 unidades. Como resultado de esta optimización, nuestro modelo alcanzó un RMSLE de 0.4925, lo que indica una mayor precisión en las predicciones de precios de las propiedades.

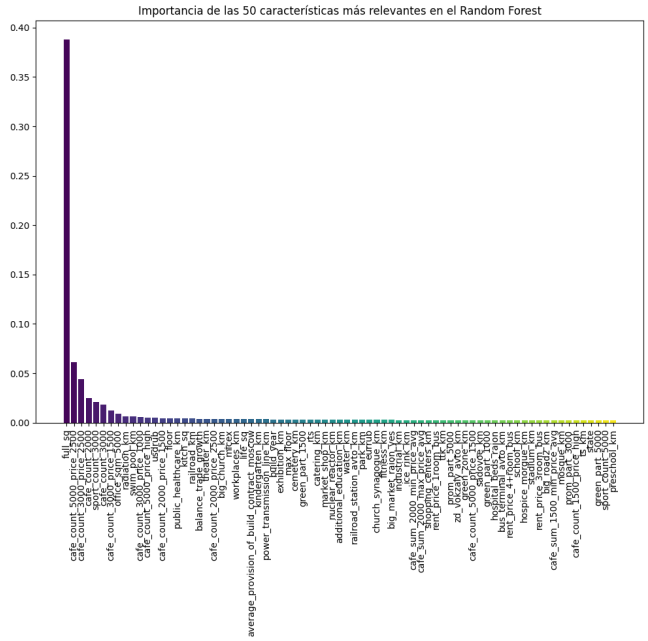
3.2.2.1 Modelo Random Forest

Comenzamos el proceso separando los datos en conjuntos de entrenamiento y prueba para validar el modelo. Realizamos un entrenamiento inicial utilizando un algoritmo de Random Forest con n_estimators=100. Los resultados arrojaron un valor de RMSLE (Error Cuadrático Medio Logarítmico) de 0.4763. Esta métrica representa una mejora significativa con respecto al modelo anterior basado en KNN, lo que sugiere que el modelo de Random Forest tiene un mejor desempeño en la predicción de precios de casas en este contexto específico. Esta diferencia en la precisión destaca la capacidad predictiva superior del modelo de Random Forest en comparación con el enfoque anterior basado en KNN.



3.2.2.2 Optimización y reducción de dimensionalidad

Iniciamos el proceso de optimización del modelo. Nuestro primer paso fue estandarizar las variables, sin embargo, no observamos mejoras en los resultados. Posteriormente, decidimos enfocarnos en las variables más significativas identificadas por el modelo de Random Forest con el fin de reducir la cantidad de características. Esta estrategia nos permitió mejorar ligeramente el desempeño del modelo, aunque con una reducción notable en la dimensionalidad de los datos.



3.2.2.3 Hiperparametrización de modelo random forest

Realizamos una hiperparametrización del modelo para mejorar su rendimiento. Dada la alta carga computacional que implica el tiempo de ejecución del modelo sin validaciones cruzadas, optamos por emplear un Randomized Search para optimizar los parámetros. Los parámetros seleccionados para esta búsqueda son los siguientes:

1. **n_estimator:** [80, 110, 140, 170, 200]
2. **max_depth:** [None, 5, 12, 19, 26, 30]
3. **min_samples_split:** [2, 5, 10]
4. **max_features:** [auto, sqrt, log2, None]
5. **criterion:** [squared_error, poisson, friendman_mse]

Esta estrategia nos permitirá explorar y seleccionar la combinación óptima de parámetros para mejorar la eficiencia y precisión del modelo, considerando diversas configuraciones sin requerir un tiempo excesivo de cómputo.

Estos fueron los resultados:

	n_estimators	max_depth	min_samples_split	min_samples_leaf	max_features	criterion	Tiempo de ejecución	RMLSE
0	108	25.0	10	2	auto	squared_error	482.776019	0.473770
1	162	21.0	5	4	None	friedman_mse	649.734655	0.474138
2	133	NaN	10	2	sqrt	squared_error	27.163838	0.479831
3	113	9.0	2	4	None	poisson	290.916056	0.482260
4	187	21.0	2	2	log2	poisson	18.701042	0.485063

3.2.3.1 Modelo XGBoost

Comenzamos dividiendo el conjunto de datos en conjuntos de entrenamiento y validación para el desarrollo del modelo XGBoost. Utilizamos los siguientes parámetros para configurar el modelo:

1. **max_depth:** 3 (Profundidad máxima del árbol)
2. **eta:** 0.1 (Tasa de aprendizaje)
3. **objective:** 'reg:squarederror' (Función de pérdida)
4. **eval_metric**: 'rmsle' (Métrica de evaluación)

Al entrenar este modelo XGBoost, obtuvimos un RMSLE de 0.4749 como resultado.
Además, es importante destacar que el modelo XGBoost mostró una notable mejora en cuanto al tiempo de ejecución en comparación con el modelo Random Forest. A pesar de obtener un RMSLE similar de 0.4749, el modelo XGBoost logró reducir significativamente el tiempo de procesamiento, lo cual es una ventaja considerable en términos de eficiencia y escalabilidad del modelo.

3.2.3.2 Optimización modelo XGBoost

La implementación de una búsqueda exhaustiva de hiperparámetros mediante Grid Search desempeñó un papel fundamental en la mejora del rendimiento predictivo del modelo XGBoost. Tras explorar

meticulosamente un espacio de hiperparámetros específicos, logramos identificar una configuración óptima que demostró ser altamente eficaz para abordar el problema planteado.

Al ajustar los parámetros del modelo XGBRegressor de XGBoost con los siguientes valores:

1. **max_depth**: 5
2. **learning_rate**: 0.1
3. **n_estimators**: 200
4. **subsample**: 1
5. **colsample_bytree**: 0.5
6. **gamma**: 0.5

Estos ajustes precisos permitieron descubrir una combinación de hiperparámetros que maximizó la precisión de las predicciones, obteniendo un valor de RMSLE de 0.468. Esta optimización resultó en una mejora significativa en la capacidad del modelo para generalizar patrones dentro de los datos, reduciendo así potenciales efectos de sobreajuste y mejorando su capacidad para realizar predicciones precisas.

max_depth	learning_rate	n_estimators	subsample	colsample_bytree	gamma	seed	Tiempo de ejecución	RMLSE
5	0.1	200	1	0.5	0.5	1	12.783116	0.468845
5	0.1	200	1	0.5	0.0	1	12.702399	0.468845
5	0.1	200	1	1.0	0.5	1	14.588423	0.469403
5	0.1	200	1	1.0	0.0	1	14.618881	0.469403
5	0.1	100	1	0.5	0.5	1	9.081825	0.471099

3.3.1 Modelos no supervisados

3.3.1.1 Red Neuronal y PCA

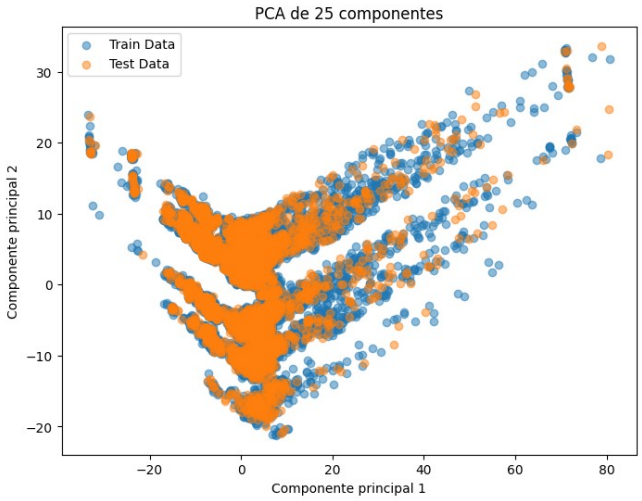
La selección de características mediante PCA (Análisis de Componentes Principales) es una técnica poderosa que reduce la dimensionalidad de los datos al transformar las variables originales en un conjunto más pequeño de componentes no correlacionados, conocidos como componentes principales. Al limitar el número de componentes a 25, se retienen las características más significativas mientras se reduce la dimensionalidad, lo que puede ayudar a simplificar y acelerar el entrenamiento de modelos, especialmente en conjuntos de datos con muchas variables originales o con multicolinealidad.

En cuanto a la arquitectura de la red neuronal, esta consta de tres capas Dense (completamente conectadas). Las capas Dense utilizan activaciones 'relu' en las capas ocultas para introducir no linealidades en el modelo y ayudar a aprender patrones complejos en los datos. La última capa tiene una sola neurona con activación 'linear', que es adecuada para problemas de regresión, como la predicción de precios de viviendas.

El modelo se compila utilizando el optimizador 'adam' y la función de pérdida 'mean_squared_error' (error cuadrático medio), comúnmente utilizada en problemas de regresión para minimizar la diferencia entre las predicciones y los valores reales.

El proceso de entrenamiento utiliza el conjunto de datos transformado por PCA (X_train_pca) para el entrenamiento, junto con las etiquetas correspondientes (y_train). Se ejecutan 10 epochs (ciclos completos a través del conjunto de datos de entrenamiento) con un tamaño de lote de 32, y se valida el modelo en el conjunto de datos de prueba (X_test_pca, y_val).

Este enfoque con PCA y red neuronal busca reducir la dimensionalidad de los datos manteniendo la información relevante, y luego emplea una red neuronal para aprender patrones complejos y realizar predicciones de manera más eficiente.



3.4 Resultados, métricas y curvas de aprendizaje

3.4.1 Resultados, métricas y curva de aprendizaje del Modelo KNN

Aplicamos una validación cruzada con un total de 25 iteraciones para evaluar la robustez y el rendimiento general del modelo en diferentes conjuntos de datos. Esto nos permitió obtener una visión más amplia y precisa del comportamiento de nuestro modelo.

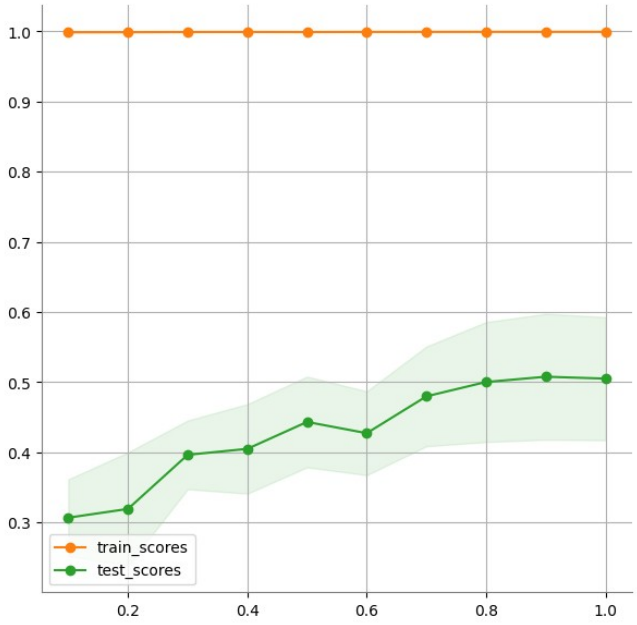
Después de las 25 validaciones cruzadas, calculamos un promedio del RMSLE, obteniendo un valor promedio de 0.4872. Además, observamos una desviación estándar de ∓0.082 en estos resultados. Esto indica que, en promedio, nuestro modelo ha demostrado un buen rendimiento en la predicción de precios de propiedades ('Price_doc').
Observaciones sobre el Rendimiento del Modelo

Sin embargo, es importante destacar que al separar los datos en 25 validaciones cruzadas, se detectó un fenómeno de sobreajuste en algunas iteraciones. Esto sugiere que el modelo puede haber aprendido demasiado de los datos de entrenamiento específicos, lo que podría limitar su capacidad para generalizar de manera efectiva con nuevos datos.

Estas observaciones resaltan la importancia de equilibrar la complejidad del modelo y su capacidad de generalización para lograr predicciones precisas y fiables en entornos de datos diversos.

Curva de aprendizaje del modelo KNN

La curva de aprendizaje revela que, a medida que incrementamos el tamaño del conjunto de datos, la precisión del modelo aumenta. Sin embargo, se observa una tendencia donde, pasando aproximadamente el 80% del tamaño del conjunto de datos, la mejora en la precisión se estabiliza. Esto sugiere que agregar más datos podría no resultar en una mejora significativa en la precisión de este modelo.



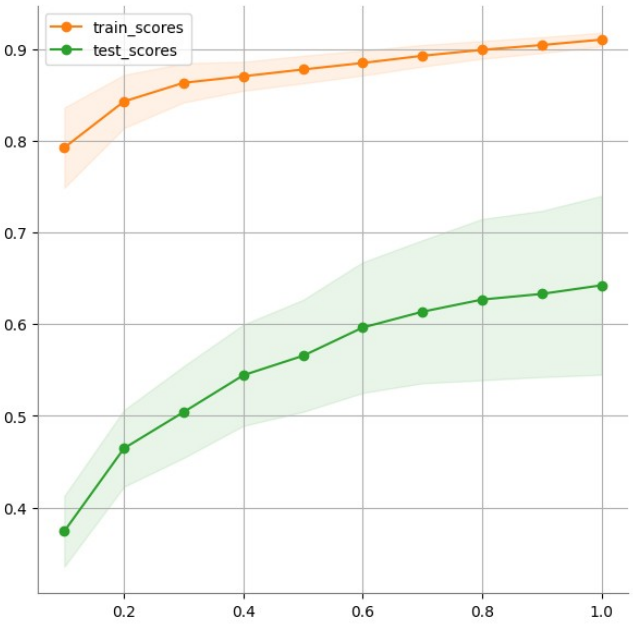
Otro aspecto crucial que la curva de aprendizaje resalta es la existencia de sobreajuste en nuestro modelo. Esta situación se evidencia por la discrepancia entre la precisión del conjunto de entrenamiento y el conjunto de validación. Mientras la precisión del conjunto de

entrenamiento se mantiene cerca del 0.99, el conjunto de validación muestra una precisión significativamente menor, alrededor del 0.5.

Esta disparidad sugiere que el modelo está aprendiendo demasiado los detalles específicos de los datos de entrenamiento, perdiendo así la capacidad de generalizar con datos nuevos o no vistos, lo que se traduce en una baja precisión en el conjunto de validación.

3.4.2 Resultados, métricas y curva de aprendizaje del Modelo Random Forest

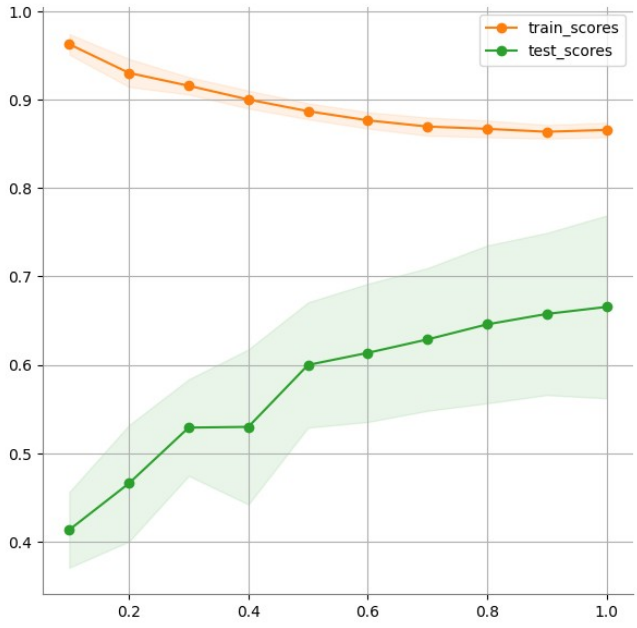
Tras realizar 5 validaciones cruzadas, obtuvimos un RMSLE promedio de 0.4706 con una desviación estándar de ±0.063, lo que indica una consistencia aceptable en el desempeño del modelo.



Al analizar la curva de aprendizaje, observamos que este modelo incrementa su precisión conforme se amplía la cantidad de datos disponibles. Sin embargo, luego de alcanzar el 80% del tamaño de los datos, la mejora en la precisión se estabiliza, lo que sugiere que la adición de más muestras probablemente no mejore considerablemente la precisión del modelo. Además, si bien se evidencia cierto sobreajuste en la curva de aprendizaje, este no es tan pronunciado como el que se observó en el modelo KNN previamente analizado.

3.4.3 Resultados, métricas y curva de aprendizaje del Modelo XGBoost

Al realizar el análisis de la curva de aprendizaje para este modelo XGBoost, notamos que su comportamiento difiere notablemente de los modelos previamente evaluados, como el KNN y el Random Forest. En particular, observamos que el modelo XGBoost exhibe menos señales de sobreajuste en comparación con los otros modelos. Obtuvimos un RMSLE de 0.468



La curva de aprendizaje reveló que a medida que se aumentaba la cantidad de datos de entrenamiento, la brecha entre las métricas de entrenamiento y validación se mantenía relativamente pequeña en el modelo XGBoost. Esto indica que el modelo no muestra una tendencia tan marcada al sobreajuste como lo hacían el KNN y el Random Forest en condiciones similares.

Este comportamiento sugiere que el modelo XGBoost tiene una mayor capacidad para generalizar patrones a partir de los datos de validación, lo que resulta en un rendimiento más estable y posiblemente más confiable al enfrentar datos no vistos.

3.4.4 Resultados y métricas de la red neuronal y PCA

A pesar de su eficiencia computacional derivada del uso de PCA, el modelo de red neuronal presentó un desempeño inferior en términos de precisión predictiva. El resultado obtenido, con un RMSLE de 0.7834, se posicionó como el peor desempeño entre todos los modelos evaluados hasta el momento.

Este resultado indica que, a pesar de la reducción de dimensionalidad aplicada con PCA, la capacidad de la red neuronal para capturar las relaciones complejas y realizar predicciones precisas sobre los precios de viviendas fue limitada en comparación con otros enfoques, como los modelos basados en árboles (Random Forest, XGBoost) y el KNN.

Esta discrepancia en la precisión podría sugerir que las características más relevantes para predecir los precios de viviendas podrían no estar completamente capturadas por los componentes principales obtenidos mediante PCA, lo que podría haber limitado la capacidad predictiva de la red neuronal en este contexto específico.

4. Retos y consideraciones de despliegue

1. Gasto Computacional Elevado: En algunos modelos, especialmente aquellos con técnicas complejas o grandes volúmenes de datos, el tiempo y recursos computacionales pueden ser excesivos, limitando su aplicabilidad en situaciones prácticas.

2. Calidad de los Datos: Los conjuntos de datos a menudo presentan inconsistencias, valores atípicos o ausencia de información relevante, lo que puede dificultar la creación de un modelo predictivo sólido. La calidad y la integridad de los datos son fundamentales para la efectividad del modelo.

3. Reducción de Dimensionalidad y Ruido en Variables: La inclusión de muchas variables, algunas de las cuales pueden no ser significativas, puede introducir ruido y complejidad innecesaria en el modelo. Estrategias como la reducción de dimensionalidad (como PCA) pueden ayudar, pero la selección de características sigue siendo un desafío crítico.

4. Elección del Mejor Modelo: En la búsqueda del modelo óptimo, se puede experimentar con varios algoritmos y técnicas. Esta variedad puede generar incertidumbre sobre cuál es el modelo más efectivo para el problema en cuestión. En ocasiones, un modelo con un rendimiento aparentemente bueno puede no ser suficiente para predecir con precisión los precios de las casas en entornos del mundo real.

Es esencial comprender que el rendimiento de un modelo puede variar según el conjunto de datos y el contexto específico. Por ejemplo, a través de un foro de discusión, se identificó que un modelo XGBoost logró un RMSLE de 0.43, lo cual parece un buen resultado. Sin embargo, en la realidad de la predicción de precios de viviendas, este valor puede indicar que el modelo aún no es lo suficientemente efectivo para realizar predicciones precisas y confiables, y podría requerir más refinamiento o consideración de otras variables o enfoques para mejorar su desempeño.

5.Conclusiones

5.1 Conclusiones del Modelo KNN

1. Optimización de Hiperparámetros: A través de métodos como Randomized Search, se logró encontrar una configuración más óptima para el modelo KNN, destacando parámetros como 'n_neighbors' = 13, 'weights' = 'Distance', 'algorithm' = 'Kd tree', entre otros.

2. Manejo de Variables: La selección de variables basada en la correlación con la variable objetivo 'Price_doc' fue esencial. Eliminar las características poco correlacionadas permitió reducir la dimensionalidad del conjunto de datos, mejorando significativamente la precisión del modelo.

3. Validación Cruzada y Sobreajuste: A través de 25 validaciones cruzadas, se observó que el modelo tenía un rendimiento promedio con un RMSLE de 0.4872, pero se identificó un sobreajuste evidente, donde la precisión en el conjunto de entrenamiento fue alta (alrededor de 0.99) mientras que en el conjunto de validación se mantuvo por debajo del 0.5.

4. Curva de Aprendizaje: La curva de aprendizaje mostró un aumento en la precisión a medida que se incrementaba el tamaño del conjunto de datos, pero después de alcanzar cierto punto, la mejora se estabilizó. Además, reveló un patrón claro de sobreajuste, indicando que el modelo no generaliza bien con nuevos datos.

5. Límites del Modelo: A pesar de los esfuerzos de optimización, la capacidad del modelo KNN parece alcanzar un límite en términos de su habilidad para mejorar la precisión, incluso con más datos. El sobreajuste detectado sugiere limitaciones en la generalización del modelo para nuevos conjuntos de datos.

En resumen, si bien el modelo KNN fue ajustado y optimizado en varios aspectos, los resultados sugieren la necesidad de considerar otros enfoques o modelos más avanzados para lograr una mayor generalización y mejorar la capacidad predictiva en la estimación de precios de propiedades ('Price_doc').

5.2 Conclusiones del modelo Random Forest

1. Desempeño del Modelo: El modelo Random Forest ha demostrado ser más efectivo en la predicción de precios de casas en comparación con el modelo KNN evaluado anteriormente. Se logró un RMSLE promedio de 0.4706 con una desviación estándar de ±0.063, lo que indica una consistencia aceptable en las predicciones.

2. Optimización y Hiperparámetros: La hiperparametrización realizada mediante un Randomized Search ayudó a encontrar una configuración óptima de parámetros, mejorando significativamente el desempeño del modelo respecto a su estado inicial.

3. Reducción de Dimensionalidad: La estrategia de reducción de dimensionalidad mediante la identificación y selección de las variables más significativas contribuyó a mejorar el modelo, aunque con una reducción en la cantidad de características.

4. Curva de Aprendizaje: La curva de aprendizaje revela que, si bien el modelo aumenta su precisión con la adición de datos, esta mejora se estabiliza después del 80% del tamaño de los datos. Además, se identificó cierto grado de sobreajuste, aunque menos pronunciado que en el modelo KNN.

5. Limitaciones y Recomendaciones: A pesar de su buen desempeño, el modelo aún muestra cierto sobreajuste y una estabilización en la mejora con más datos.

Se recomienda explorar técnicas adicionales para reducir el sobreajuste y considerar otras estrategias para mejorar la precisión.

5.3 Conclusiones del modelo XGBoost

1. Desempeño Comparativo: El modelo XGBoost demostró una capacidad comparable al Random Forest en términos de precisión, obteniendo un RMSLE de 0.4749 en su configuración inicial. Aunque similar en términos de métrica, logró reducir significativamente el tiempo de ejecución en comparación con el Random Forest.

2. Optimización de Hiperparámetros: La búsqueda exhaustiva de hiperparámetros mediante Grid Search permitió identificar una combinación óptima de valores. La configuración refinada produjo un RMSLE de 0.468, mejorando la capacidad del modelo para generalizar patrones y reducir el sobreajuste.

3. Curva de Aprendizaje: A diferencia de otros modelos como el KNN y el Random Forest, el XGBoost mostró una curva de aprendizaje con una brecha menor entre las métricas de entrenamiento y validación. Esto sugiere una capacidad superior para generalizar y una menor tendencia al sobreajuste.

4. Eficiencia en Tiempo de Ejecución: Además de su buen desempeño, el modelo XGBoost resalta por su eficiencia en términos de tiempo de procesamiento, lo que lo convierte en una opción viable para conjuntos de datos extensos.

5.4 Conclusiones de la Red Neuronal y PCA

1. Limitaciones de la Red Neuronal con PCA: Aunque PCA redujo la dimensionalidad del conjunto de datos, posiblemente no capturó todas las características fundamentales necesarias para una predicción precisa de los precios de viviendas. La limitación en la representación de las características más importantes puede haber impactado negativamente en la capacidad predictiva de la red neuronal.

2. Inferioridad en Precisión: La precisión obtenida fue considerablemente más baja en comparación con otros modelos, lo que sugiere que la red neuronal no logró generalizar los patrones presentes en los datos de manera tan efectiva como otros enfoques, como los modelos basados en árboles o vecinos más cercanos.

3. Relevancia de las Características Seleccionadas: Es posible que la selección de 25 componentes principales a través de PCA no haya capturado adecuadamente todas las relaciones complejas y relevantes entre las características originales, lo que pudo haber limitado el poder predictivo de la red neuronal.

5.5 Conclusiones generales

En resumen, tras un exhaustivo análisis de modelos para la predicción de precios de viviendas, se determinó que el XGBoost sobresalió como el mejor modelo. Mostró una capacidad más robusta para generalizar y menor sobreajuste en comparación con otros modelos evaluados, lo que indicaba una mejor capacidad de adaptación a datos no vistos.

Sin embargo, a pesar de su mejor rendimiento en el contexto del conjunto de datos disponible, se concluyó que no sería recomendable implementar el modelo XGBoost en un entorno de despliegue práctico. La calidad de los datos existentes no proporcionó la precisión suficiente en las predicciones, lo que generó un margen de error significativo y no brindó estimaciones de precios lo suficientemente cercanas a la realidad para satisfacer las expectativas de los clientes o usuarios finales.

Esta evaluación subraya la importancia crítica de la calidad de los datos en el éxito de los modelos predictivos, ya que incluso el mejor modelo puede enfrentar desafíos significativos si la calidad de los datos subyacentes es limitada. Por lo tanto, se sugiere una revisión y mejora de la calidad de los datos antes de considerar la implementación de modelos predictivos en situaciones del mundo real.