Entrega Final de proyecto

POR:

Cristian David Tamayo Espinosa

MATERIA:

Introducción a la inteligencia artificial

PROFESOR:

Raul Ramos Pollan

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

MEDELLÍN 2023

Introducción:

1.1 Planteamiento del problema

El problema predictivo que se desea resolver consiste en predecir los precios de viviendas a partir de numerosas características diferentes. Estas características incluyen información sobre el tamaño del lote, la calidad de la fachada, el número de habitaciones y baños, y la ubicación geográfica, entre otras.

El objetivo es desarrollar un modelo de regresión que pueda predecir con precisión el precio de una casa en base a estas características. Para ello, se cuenta con un conjunto de datos de entrenamiento que incluye información sobre viviendas y sus precios. Además, se dispone de otro conjunto de datos de prueba que incluye información sobre viviendas, pero sin el precio.

Por lo que el objetivo final de este problema es ayudar a los compradores, vendedores y corredores de bienes raíces a tomar decisiones informadas sobre el valor de las propiedades. Además de que al predecir con precisión el precio de una casa, se puede obtener una estimación del valor real de la propiedad y evitar transacciones injustas para ambas partes.

2. Dataset

El conjunto de datos a utilizar proviene de una competición de Kaggle en la que se proporcionan datos de propiedades inmobiliarias en Ames, Iowa, Estados Unidos.

El conjunto de datos consta de cuatro archivos: train.csv, test.csv, data_description.txt y sample_submission.csv.

El archivo **train.csv** es el conjunto de datos de entrenamiento y contiene información sobre 1,460 propiedades

Mientras que el archivo **test.csv** es el conjunto de datos de prueba y contiene información sobre 1,459 propiedades.

El archivo **data_description.txt** proporciona una descripción completa de cada columna en los archivos de datos

El archivo **sample_submission.csv** es una presentación de referencia de una regresión lineal en el año y mes de venta, el tamaño del lote y el número de habitaciones.

Los campos de datos que incluyen que podremos encontrar tanto en train.csv como test.csv

SalePrice: El precio de venta de la propiedad en dólares. Esta es la variable objetivo que se intenta predecir.

MSSubClass: La clase de construcción

MSZoning: La clasificación general de zonificación

LotFrontage: Pies lineales de calle conectados a la propiedad

LotArea: Tamaño del lote en pies cuadrados

Street: Tipo de acceso a la carretera

Alley: Tipo de acceso a callejón

LotShape: Forma general de la propiedad

LandContour: Plano de la propiedad

Utilities: Tipo de servicios públicos disponibles

LotConfig: Configuración del lote

LandSlope: Pendiente de la propiedad

Entre muchos otros datos más que encontraremos allí adentro.

3. métricas

Las métricas de desempeño para evaluar las predicciones en esta competición se basan en el cálculo del error cuadrático medio (RMSE) entre el logaritmo de los valores de venta observados y el logaritmo de los valores de venta pronosticados. Tomar los logaritmos garantiza que los errores en la predicción de casas caras y baratas afecten igualmente el resultado final. En resumen, se busca minimizar el RMSE en la escala de logaritmos para obtener las predicciones más precisas posibles.

4. Desempeño

un primer criterio deseable de desempeño en producción podría ser que el modelo sea capaz de predecir con precisión los precios de venta de las viviendas con un bajo valor de RMSE, es decir, cuanto menor sea el valor de RMSE obtenido por el modelo, mejor será su desempeño en producción.

Ya que con un bajo valor de RMSE, se puede inferir que el modelo de regresión utilizado para predecir los precios de las casas es más preciso en sus predicciones.

2 exploración de variables

Lo primero que se realizó fue importar las bibliotecas necesarias, incluyendo numpy, pandas, matplotlib y seaborn, ya con ellas leímos los datos del conjunto de entrenamiento y del conjunto de prueba utilizando el método **pd.read_csv**.

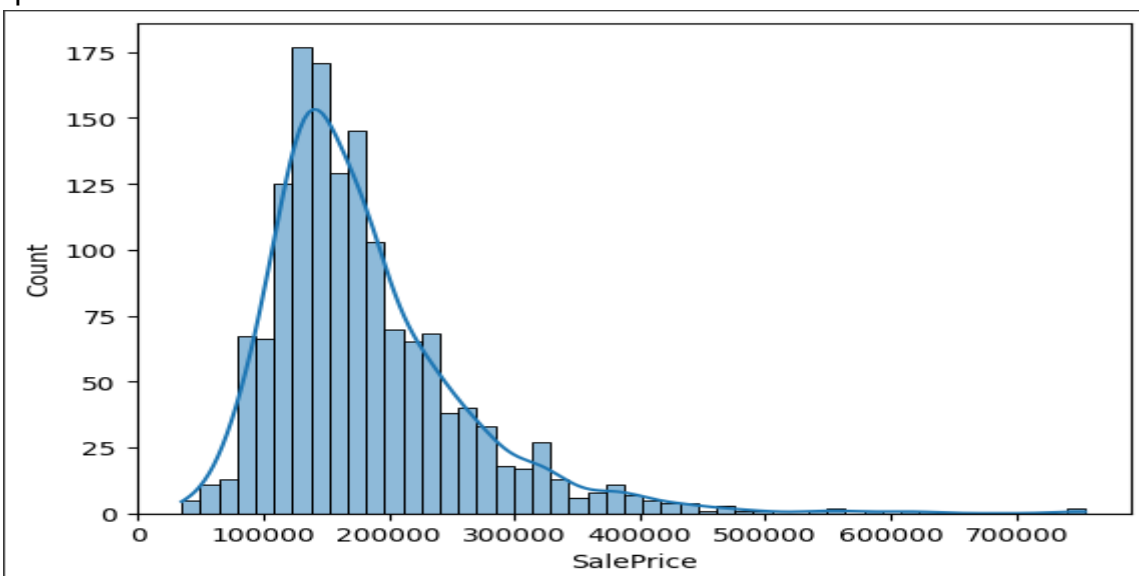
Una vez cargados nuestros datos en el entorno de colab se realiza una primera exploración de los datos, para ello utilicé el método **data.info()** para obtener una visión general de los datos y verificar si existen valores nulos en ellos, además se logró identificar que hay 35 características con valores enteros, 3 características de valores flotante y 43 características con valores object.

Una vez observado esto, me dispongo a utilizar el método **data.describe()** el cual me proporcionó estadísticas descriptivas de los datos numéricos del dataset, de esta manera pude conocer:

- El número de elementos no nulos (count)
- La media (mean)
- La desviación estándar (std)
- El valor mínimo (min)
- El percentil 25 (25%)
- El percentil 50, también conocido como la mediana (50%)
- El percentil 75 (75%)
- El valor máximo (max)

2.1 Distribución de la variable objetivo

Posterior a ello realice una exploración a la distribución de cómo se veía y comportaba de una manera estadística la variable objetivo que, en mi caso, es la variable del precio de venta “sale Price”. El cual gráficamente se puede observar que no tiene una distribución normal:



2.2 Datos faltantes

Otra de las cuestiones de vital importancia a la hora de la exploracion del dataset, fue tener un conocimiento solido acerca de cuanto podria ser la cantidad de datos faltantes, esto podria tener un impacto significativo en el rendimiento de los modelos de predicción una vez ya esten establecidos, o durante el funcionamiento y ejecucion de los mimos.

Observemos a continuacion cuantos datos faltantes hay por categoria:

```
LotFrontage      259
Alley           1369
MasVnrType         8
MasVnrArea         8
BsmtQual          37
BsmtCond          37
BsmtExposure      38
BsmtFinType1      37
BsmtFinType2      38
Electrical         1
FireplaceQu      690
GarageType        81
GarageYrBlt       81
GarageFinish      81
GarageQual        81
GarageCond        81
PoolQC          1453
Fence           1179
MiscFeature     1406
dtype: int64
```

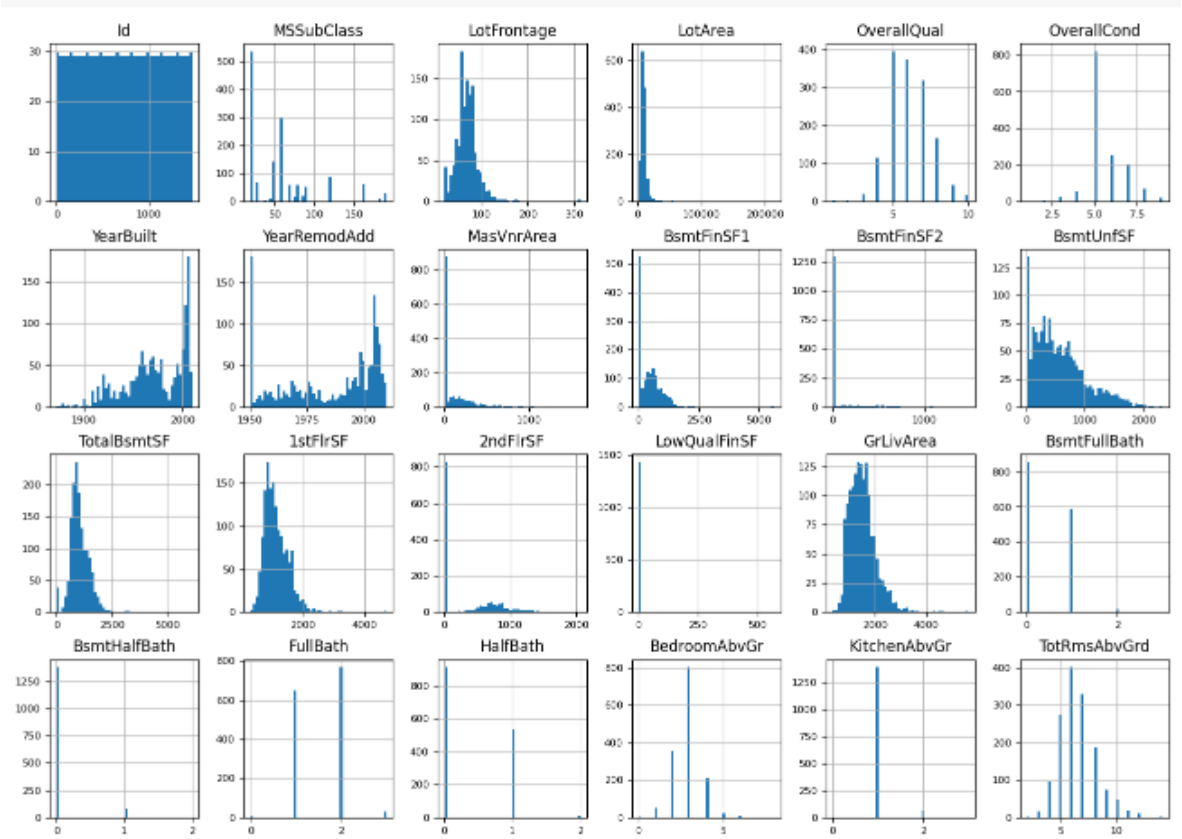
Estos valores ausentes pueden introducir sesgos y afectar la calidad de las predicciones resultantes. Por ejemplo, si la categoria "PoolQC" tiene 259 datos faltantes, es posible que falte información crucial para determinar el precio de venta de una casa y que realmente no brinde tanta informacion.

Por lo tanto, gracias a esa obervacion y exploracion, se da riendas a estar al tanto de esto, para mas adelante en la realizacion del preprocesado de los datos, estos datos faltantes sean tratados de una manera adecuada, ya que el tratado adecuado de los mismo es necesario para evitar inconvenientes en los resultados que se esperen de los modelos que se piensen elaborar e implementar.

Po lo que es fundamental comprender el dominio del problema, que en nuestro caso es el precio de venta de las casas y la importancia de cada categoria antes de decidir cómo tratar los valores faltantes.

2.3 Distribucion de las categoricas numéricas.

Dentro de la exploracion de los datos fue util conocer como se distribuyen las variables categorias numericas, ya que esta representación visual puede ser útil para varios análisis y proporciona información valiosa sobre las características de los datos.



Al observar la distribución de las categorías numéricas se logro identificar la siguiente informacion:

forma de distribución: Los histogramas muestran cómo se distribuyen los valores en cada categoría numérica. Ayudando identificar si la distribución es simétrica , hacia la izquierda o derecha, o si presenta múltiples picos. Esta información me ayudó a comprender mejor la naturaleza de los datos y ajustar el análisis.

Detectar valores atípicos: se logra identificar la presencia de valores atípicos o anomalías en los datos. Estos valores inusuales pueden afectar negativamente los modelos de predicción y los análisis estadísticos.

Evaluar la dispersión de los datos: La forma y la amplitud de la distribución en el histograma me pudieron proporcionar información sobre la dispersión de los datos. Una distribución más ancha indica una mayor variabilidad, mientras que una distribución más estrecha sugiere una menor dispersión.

Identificacion de relaciones entre variables: Al haber múltiples categorías numéricas, se logra identificar patrones o relaciones entre ellas.

3. Iteraciones de desarrollo.

Tratamiento de datos

3.1 Eliminacion de caracteristicas que sólo se centraban en una categoría.

Durante el desarrollo de mi proyecto, me encontré con un conjunto de datos que contenía varias características que se centraban únicamente en una categoría. Para abordar esta situación, decidí eliminar estas características del conjunto de datos. El objetivo principal de esta acción fue reducir la redundancia y mejorar la eficiencia y efectividad de mi análisis, para una vez con la implementacion de los modelos, se logre generar resultados de una manera mucho mas optima.

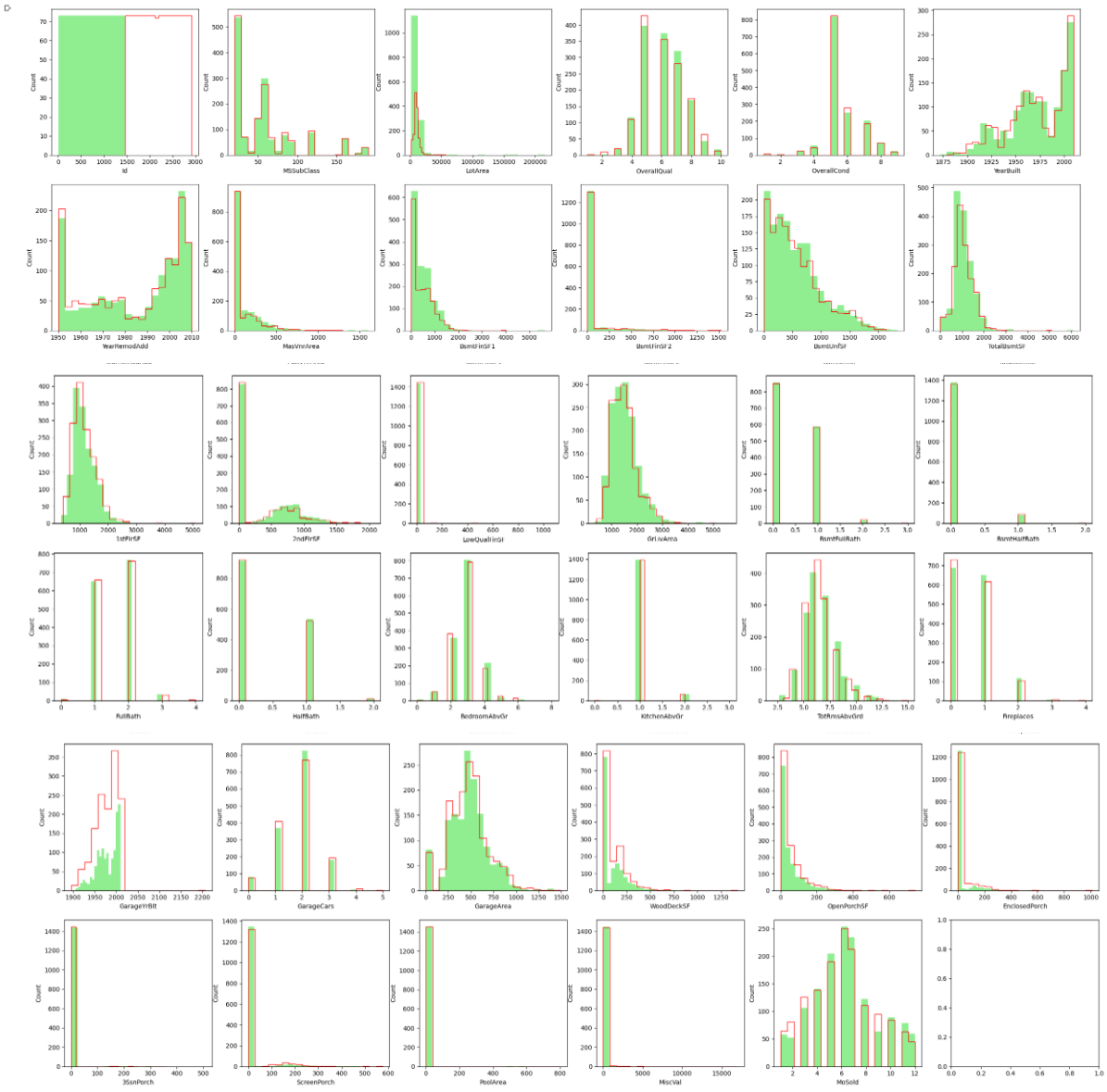
La eliminación de esas características resultó útil por las siguientes razones:

1. Reducción de la dimensionalidad: Al eliminar las características que solo contenían una categoría constante, reduje la dimensionalidad del conjunto de datos. Esto es especialmente importante cuando se trabaja con un gran número de características, ya que una alta dimensionalidad puede llevar a un aumento en la complejidad y dificultades en el análisis y la interpretación de los resultados. Al eliminar características redundantes, mejoré la eficiencia computacional y facilité la comprensión de los datos.
2. Eliminación de información irrelevante: Al eliminar estas características, me deshice de información irrelevante que no contribuía de manera significativa al objetivo de mi proyecto. Esto me ayudó a enfocarme en las características relevantes y mejoró la calidad de los resultados obtenidos.
3. Mejora de la estabilidad del modelo: La presencia de características constantes puede introducir ruido innecesario en los modelos de aprendizaje automático y afectar negativamente su rendimiento. Al eliminar estas características, evité que los modelos se sobreajustaran a valores constantes y mejoré su estabilidad y generalización en datos nuevos.

3.2 analisis de los datos continuos

Con el analisis de los datos continuos se pretende justicar y encontrar las razones por las cuales resultó necesario eliminar ciertas categorias que no resultan utiles para nuestro analisis.

las barras verdes representan los datos de entrenamiento y las rojas los datos de prueba.



Los gráficos anteriores muestran que hay valores atípicos en múltiples categorias, mientras que algunas categorias no son útiles, por lo que resulta necesario tratar esos valores atipicos, y a su vez tratar los datos que no resultan utiles.

3.3 Tratando los valores atipicos

A Continuacion se expondra como fue el abordaje de los valores atipicos que me encontré dentro del conjunto de datos. No obstante, cabe descatar las razones por los cuales se opto por tratar esos valores atipicos, pues, estos valores son observaciones que se alejan significativamente del patrón general de los datos y pueden tener un impacto negativo en el análisis y posteriormente en los modelos.

Por lo tanto, decidí abordar esta situación mediante la eliminación de los valores atípicos utilizando el rango intercuartil como criterio.

El codigo en el cual se hace uso del rango intercuartil es el siguiente:

```
[ ]  # HAciendo uso del rango intercuartil
     def valorAtipico(data_temp, k=3):
         q1, q3 = np.percentile(data_temp, [25, 75])

         iqr = q3 - q1
         lower_bound = q1 - (k * iqr)
         upper_bound = q3 + (k * iqr)

         valorAtipico = [index for index,x in enumerate(data_temp) if x < lower_bound or x > upper_bound]

         return valorAtipico
```

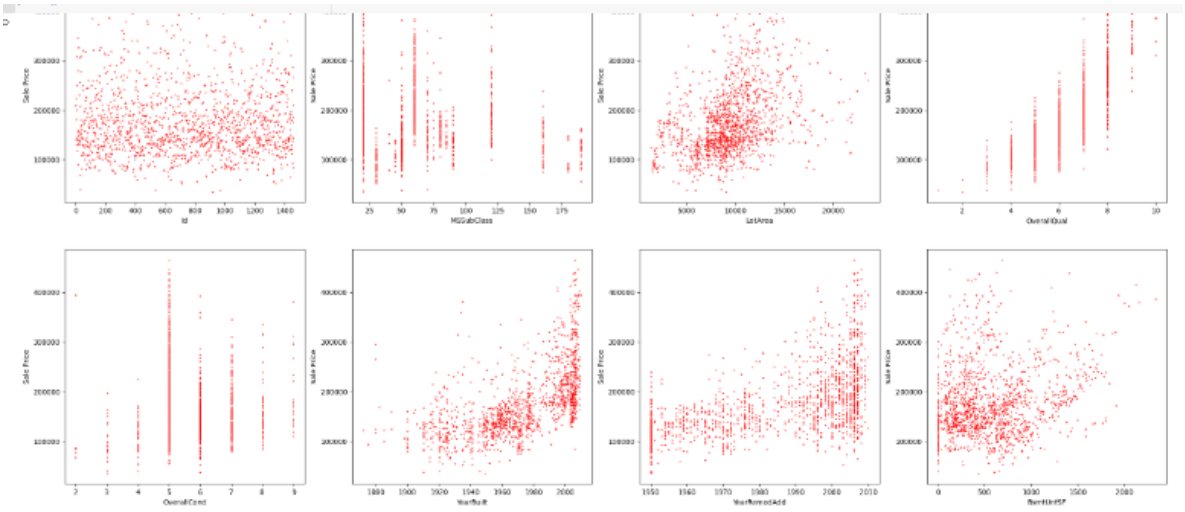
A la hora de comprender los valores atípicos se llegó a la conclusion de que pudieron ser el resultado de errores de medición, ingreso incorrecto de datos o casos extremadamente raros. Estos valores inusuales pueden distorsionar la distribución de los datos y afectar la precisión de los análisis y los modelos. Al eliminar los valores atípicos, me aseguré de que los datos utilizados fueran más confiables y representativos del comportamiento general del conjunto de datos.

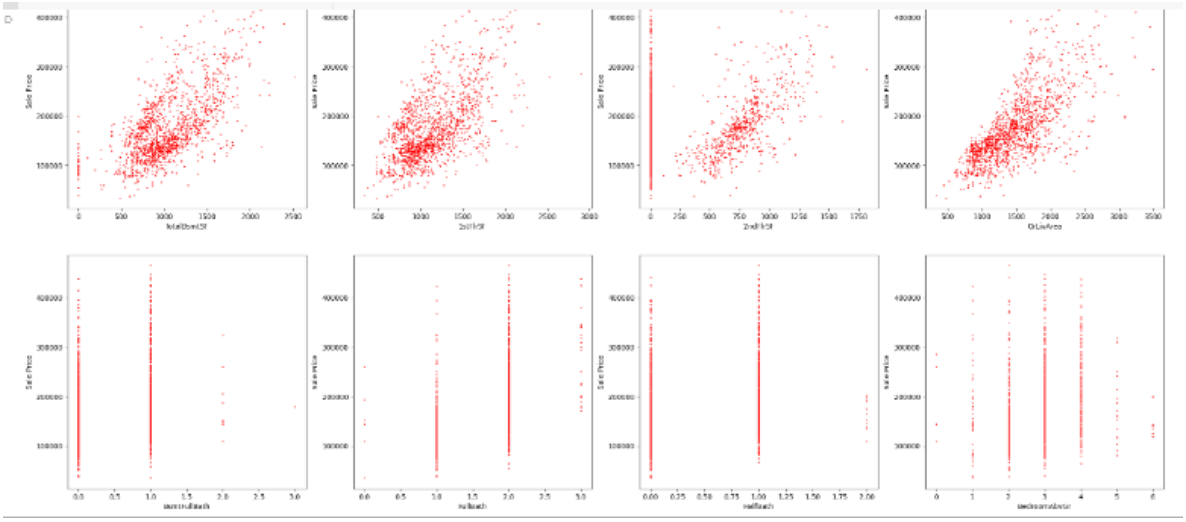
3.4 Datos continuos vs Precio de venta

Durante el tratamiento de los datos tambien resulta util realizar un analisis con respecto a la comparacion de los datos continuos vs precio de venta, ya que esto permite comprender cómo las características continuas del conjunto de datos están relacionadas con el precio de venta. Al graficar la dispersión de estas características con respecto al precio de venta, podemos identificar posibles patrones, tendencias o relaciones lineales entre ellas. Esto nos brinda información

valiosa sobre qué características pueden influir significativamente en el precio de venta de una propiedad y cuáles pueden tener un impacto menor.

Y del mismo modo nos da una idea de como se comportan los datos dependientes e independientes.





3.5 Separando datos dependientes e independientes

La separacion de datos dependientes e independientes se hace con el objetivo de distinguir las variables predictoras (independientes) de la variable objetivo (dependiente) en el conjunto de datos. La razón principal detrás de esta separación es poder una vez con los modelos predictivos entrenarlos o realizar

análisis estadísticos utilizando las variables independientes para predecir o explicar la variable dependiente.

La separación de los datos dependientes e independientes no afecta directamente los datos en sí. Lo que se logra con esto es dividir el conjunto de datos en dos partes: una matriz de características (X) que contiene todas las variables independientes, y un vector objetivo (y) que contiene los valores de la variable dependiente.

El codigo con el que se realizó esto es el siguiente:

```
[ ]  X = data_temp.iloc[:, :-1]
     y = data_temp['SalePrice']
```

3.6 Tratando los Datos nulos

Durante mi análisis del conjunto de datos, identifiqué la presencia de datos nulos o faltantes en algunas variables. Los datos nulos pueden afectar la calidad y confiabilidad de nuestros análisis y modelos. Por lo tanto, abordar esta situación se volvió crucial para garantizar la integridad de nuestros resultados.

El codigo utilizado fue el siguiente:

```
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=10)
X = imputer.fit_transform(X)
test_data = imputer.transform(test_data)
```

3.7 Partición de los datos

Esto con el fin de evaluar la capacidad de generalización de los modelos que voy a implementar. Al separar el conjunto de datos en dos conjuntos distintos, uno para entrenamiento y otro para prueba, podemos medir qué tan bien se desempeñarian los modelos.

El código que utilicé divide los datos en cuatro variables: xtrain, xtest, ytrain y ytest. Estas variables contienen las características (variables independientes) y el objetivo (variable dependiente) respectivamente, divididos en conjuntos de entrenamiento y prueba.

El conjunto de entrenamiento (xtrain y ytrain) se utiliza para ajustar el modelo, es decir, para aprender los patrones y relaciones en los datos. El conjunto de prueba (xtest y ytest) se utiliza para evaluar el rendimiento del modelo en datos no vistos durante el entrenamiento.

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(X,y, test_size=0.2, random_state=1)
```

3.8 Normalizando las variables

Al normalizar las variables lo hago con el fin de estandarizar o normalizar las escalas de las características, lo cual puede tener varios beneficios en el análisis y modelado de datos, tales como:

Eliminación del sesgo debido a las diferentes escalas, Mejora del rendimiento de los modelos, eliminar los sesgos debidos a las diferentes escalas y se facilita el análisis y modelado de datos de manera más precisa y eficiente.

El codigo para ello fue el siguiente:

```
scaler = StandardScaler()

xtrain = scaler.fit_transform(xtrain)
xtest = scaler.transform(xtest)

test_data1 = scaler.transform(test_data)
```

3.9 reduccion de informacion irrelevante:

Mediante la reducción de dimensionalidad, logro reducir la cantidad de variables o características del dataset manteniendo la mayor cantidad posible de información relevante.

Lo cual se hace con el siguiente codigo:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=10)

xtrain = pca.fit_transform(xtrain)
xtest = pca.transform(xtest)

test_data1 = pca.transform(test_data1)
```

4 Modelos

Linear Regression:

- Algoritmo supervisado: Sí
- Hiperparámetros: {'linearregression__copy_X': True, 'linearregression__fit_intercept': True}
- Resultado: Accuracy Score = 0.8585425395628915

El modelo de regresión lineal utiliza una función lineal para predecir los valores de salida. En este caso, se utilizó la implementación de regresión lineal de la librería Scikit-learn. Se exploraron dos hiperparámetros: **fit_intercept** (para ajustar o no el término de intercepción) y **copy_X** (para copiar o no los datos de entrada). El mejor puntaje de precisión obtenido fue de 0.8585.

Decision Tree:

- Algoritmo supervisado: Sí
- Hiperparámetros: {'clf2__max_depth': 5, 'clf2__max_leaf_nodes': 20, 'clf2__min_samples_leaf': 4, 'clf2__min_samples_split': 2}
- Resultado: Accuracy Score = 0.8228785744510577

El modelo de árbol de decisión es otro algoritmo de aprendizaje supervisado. Este modelo divide repetidamente los datos en función de las características para formar una estructura en forma de árbol. Se realizaron ajustes en los hiperparámetros, como la profundidad máxima del árbol, el número mínimo de muestras requeridas para dividir un nodo y el número máximo de nodos hoja permitidos. El mejor puntaje de precisión obtenido fue de 0.8229.

Random Forest:

- Algoritmo supervisado: Sí
- Hiperparámetros: {'clf3__max_depth': 15, 'clf3__max_features': 'auto', 'clf3__min_samples_leaf': 3, 'clf3__min_samples_split': 2, 'clf3__n_estimators': 200}
- Resultado: Accuracy Score = 0.8773441510796779

El modelo de Bosque Aleatorio es un ensamblaje de múltiples árboles de decisión. Cada árbol se entrena en una muestra aleatoria de datos y las predicciones se promedian para obtener la predicción final. Se exploraron diferentes hiperparámetros, como el número de estimadores (árboles) en el bosque, la profundidad máxima de los árboles, el número mínimo de muestras requeridas para dividir un nodo, entre otros. El mejor puntaje de precisión obtenido fue de 0.8773.

Comparación de modelos: Se realiza una comparación de los tres modelos utilizando el puntaje de precisión. En el gráfico se muestra la puntuación obtenida durante el ajuste de hiperparámetros (en naranja) y la puntuación obtenida en los datos de prueba (en verde claro). Según la gráfica, el modelo de Bosque Aleatorio obtuvo el puntaje de precisión más alto, seguido por la Regresión Lineal y el Árbol de Decisión.

Final Model: El modelo final seleccionado es el modelo de Bosque Aleatorio (Random Forest), con los mejores hiperparámetros encontrados. Se utiliza este modelo para realizar las predicciones en los datos de prueba (**test_data1**) y se almacenan en la variable **ypredf**.