1. Relación lineal: Si hay una línea clara y definida en un gráfico de dispersión, indica una relación lineal entre esas dos características. Por ejemplo, si el gráfico de dispersión de "area\_m2" vs. "prices" muestra una línea ascendente, significa que a medida que aumenta el área en metros cuadrados, también tiende a aumentar el precio.
2. Correlación: Observa la forma general de los gráficos de dispersión. Si los puntos tienden a agruparse alrededor de una forma específica (por ejemplo, en forma de U, S o elipse), indica una relación no lineal o una correlación compleja entre las características.
3. Distribución: Examina los histogramas en la diagonal. Te permiten comprender cómo se distribuyen las características individualmente. Por ejemplo, si el histograma de "rooms" muestra una distribución sesgada hacia la derecha, significa que hay más viviendas con un mayor número de habitaciones.
4. Outliers: Presta atención a los puntos que se desvían significativamente de la tendencia general en los gráficos de dispersión. Estos puntos podrían indicar valores atípicos o anómalos en los datos.

En resumen, el pair plot te ayuda a comprender visualmente la relación y la distribución de las características seleccionadas. Puedes utilizar estas observaciones para obtener información sobre cómo las características se relacionan entre sí y cómo influyen en la variable objetivo, en este caso, el precio de las viviendas.

Este código generará un gráfico de dispersión que muestra la relación entre los precios reales y los precios predichos por el modelo de regresión. Cada punto en el gráfico representa una observación, y la posición del punto muestra el valor real en el eje x y el valor predicho en el eje y. Esto te permitirá visualizar cómo se comparan los valores predichos con los valores reales y evaluar la calidad del modelo.

Este código generará un gráfico de dispersión que muestra la relación entre las predicciones del modelo y los residuos. Los residuos son la diferencia entre los valores reales y los valores predichos por el modelo. El eje x representa las predicciones del modelo, mientras que el eje y representa los residuos. Esto te permitirá evaluar la distribución y patrones de los residuos, lo cual es importante para verificar si el modelo de regresión está capturando correctamente la variabilidad de los datos.

El mensaje que estás viendo es una descripción de los parámetros y opciones disponibles para el objeto **SVR** en el paquete **sklearn.svm**. Te proporciona información sobre los diferentes parámetros que puedes ajustar al crear un objeto **SVR** y su significado.

Aquí hay una explicación de algunos de los parámetros mencionados en el mensaje:

* **C**: Es el parámetro de penalización del término de error. Controla el equilibrio entre una frontera de decisión suave y la correcta clasificación de los puntos de entrenamiento.
* **kernel**: Especifica el tipo de hiperplano utilizado para separar los datos. Puede ser uno de los siguientes: 'linear' (lineal), 'poly' (polinomial), 'rbf' (función de base radial), 'sigmoid' (sigmoide), 'precomputed' (precalculado) o una función llamable.
* **degree**: Es el grado de la función de kernel polinómico ('poly'). Este parámetro se ignora en otros tipos de kernel.
* **gamma**: Es un parámetro para hiperplanos no lineales. Cuanto mayor sea el valor de gamma, más intentará ajustar exactamente el conjunto de datos de entrenamiento. El valor predeterminado actual es 'auto', que utiliza 1 / n\_features.
* **coef0**: Es un término independiente en la función de kernel. Solo es significativo en los kernel 'poly' (polinómico) y 'sigmoid' (sigmoide).
* **shrinking**: Es un indicador booleano que determina si se utiliza la heurística de reducción de muestras. La reducción de muestras puede acelerar el tiempo de entrenamiento en conjuntos de datos grandes, pero no afecta los resultados del modelo.

Estos parámetros te permiten ajustar el comportamiento del modelo **SVR** según tus necesidades y el tipo de datos con los que estés trabajando. Puedes modificar estos parámetros al crear el objeto **SVR** para experimentar y obtener mejores resultados en tu problema de regresión.

El gráfico que se muestra después de ejecutar el código representa la relación entre los precios reales y los precios predichos por el modelo de regresión de vectores de soporte (SVM).

Cada punto en el gráfico representa una muestra del conjunto de entrenamiento. El eje x representa los precios reales y el eje y representa los precios predichos por el modelo. Idealmente, si el modelo predice los precios de manera perfecta, todos los puntos se alinearían en una línea diagonal de pendiente 1.

Sin embargo, en la mayoría de los casos, existirán diferencias entre los precios reales y los precios predichos. El gráfico nos permite visualizar estas diferencias. Si los puntos están dispersos alrededor de la línea diagonal, significa que el modelo tiene cierta capacidad predictiva. Si los puntos están muy dispersos y no siguen una tendencia clara, puede indicar que el modelo no está capturando adecuadamente los patrones en los datos.

El objetivo es minimizar las diferencias entre los precios reales y los precios predichos, lo que implicaría puntos cercanos a la línea diagonal. Si el gráfico muestra un patrón claro de dispersión o tendencia, esto puede indicar áreas en las que el modelo necesita mejorar o ajustarse para lograr mejores predicciones.

En resumen, el gráfico permite evaluar visualmente cómo se ajustan las predicciones del modelo a los valores reales y proporciona una indicación de la calidad del ajuste del modelo.

El gráfico que se muestra con el código proporcionado es un gráfico de dispersión que representa la relación entre los valores predichos (**y\_pred**) y los residuos (**y\_train - y\_pred**) en un problema de regresión.

* En el eje x (horizontal), se encuentran los valores predichos (**y\_pred**). Estos valores representan las predicciones realizadas por el modelo de regresión.
* En el eje y (vertical), se encuentran los residuos. Los residuos son la diferencia entre los valores reales de la variable objetivo (**y\_train**) y los valores predichos (**y\_pred**). Los residuos representan el error o la discrepancia entre las predicciones del modelo y los valores reales.

El gráfico de dispersión permite visualizar la relación entre las predicciones y los residuos. En un escenario ideal, no debería haber un patrón discernible en los residuos y deberían distribuirse de manera aleatoria alrededor de cero. Esto indicaría que el modelo de regresión es capaz de capturar la estructura y la tendencia de los datos de manera adecuada.

Al observar el gráfico, es importante verificar si hay algún patrón o tendencia en la distribución de los residuos. Si los residuos muestran un patrón sistemático (por ejemplo, una forma de U o una curva), esto puede indicar que el modelo tiene problemas para capturar ciertas características de los datos o que existe una relación no lineal entre las variables predictoras y la variable objetivo.

Además, también es importante verificar si la dispersión de los residuos es constante en todo el rango de valores predichos. Si hay una variación en la dispersión de los residuos (es decir, la dispersión de los residuos aumenta o disminuye a medida que aumentan o disminuyen los valores predichos), esto puede indicar heterocedasticidad, lo que sugiere que el modelo no se ajusta de manera óptima a los datos.

En resumen, el gráfico de dispersión "Predicted vs Residuals" proporciona información sobre la calidad del modelo de regresión, la existencia de patrones sistemáticos o no linealidades en los datos y la heterocedasticidad. Es una herramienta útil para evaluar el rendimiento del modelo y detectar posibles mejoras o problemas en el ajuste del modelo a los datos.

1. Compilación del modelo: Compila el modelo especificando el optimizador, la función de pérdida y las métricas que se utilizarán durante el entrenamiento. Por ejemplo, puedes utilizar el optimizador Adam, la función de pérdida de error cuadrático medio (MSE) y la métrica de error absoluto medio (MAE).
2. Entrenamiento del modelo: Ajusta el modelo a los datos de entrenamiento utilizando el método fit. Especifica el número de épocas (iteraciones) y el tamaño del lote (batch size) para el entrenamiento.
3. Evaluación del modelo: Evalúa el rendimiento del modelo utilizando los datos de prueba. Calcula métricas como el MSE y el MAE para evaluar qué tan bien se ajusta el modelo a los datos de prueba.

# Importamos las librerías que vamos a necesitar

import numpy as np

import pandas as pd

import os

from sklearn.ensemble import RandomForestRegressor

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix, classification\_report

data\_url = r"C:\Users\Ferran\OneDrive - es.logicalis.com\Escritorio\hkgn\hkgn pf\Barcelona\_Idealista.csv"

if os.path.exists(data\_url):

raw\_df = pd.read\_csv(data\_url, sep=";")

# Resto del código...

else:

print("El archivo no existe en la ubicación especificada.")

print(raw\_df.head(10))

district rooms area\_m2 lift prices

0 Eixample 3 351 Yes 2300000

1 Sarrià-Sant Gervasi 5 298 Yes 1750000

2 Eixample 2 90 Yes 590000

3 Eixample 2 130 Yes 450000

4 Horta-Guinardó 3 110 No 430000

5 Eixample 5 161 Yes 690000

6 Eixample 2 110 No 395000

7 Eixample 3 106 Yes 595000

8 Les Corts 2 62 Yes 450000

9 Ciutat Vella 5 152 Yes 665000

raw\_df = pd.read\_csv(data\_url, sep=";")

num\_filas, num\_columnas = raw\_df.shape

print("Número de filas:", num\_filas)

print("Número de columnas:", num\_columnas)

Número de filas: 3265

Número de columnas: 5

data = pd.read\_csv(data\_url, sep=";")

resumen\_estadistico = data.describe()

print(resumen\_estadistico)

rooms area\_m2 prices

count 3265.000000 3265.000000 3.265000e+03

mean 3.160796 121.566616 6.004200e+05

std 1.325257 92.721515 6.522661e+05

min 1.000000 20.000000 4.900000e+04

25% 2.000000 72.000000 2.799000e+05

50% 3.000000 93.000000 4.190000e+05

75% 4.000000 135.000000 6.550000e+05

max 13.000000 900.000000 1.200000e+07

# Reemplaza 'data' con el nombre de la variable que contiene tus características

X = data[['area\_m2', 'prices']] # Selecciona las columnas 'area\_m2' y 'prices'

# Reemplaza 'data' con el nombre de la variable que contiene tus etiquetas

y = data['district'] # Selecciona la columna 'district' como etiqueta

# Utiliza la función train\_test\_split para dividir los datos en conjuntos de entrenamiento y prueba

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,

y,

test\_size=0.2,

random\_state=4)

# Imprime las formas de los conjuntos de entrenamiento y prueba

print('Entrenamiento y prueba: {} {}'.format(X\_train.shape, X\_test.shape))

Entrenamiento y prueba: (2612, 2) (653, 2)

data = pd.read\_csv(data\_url, sep=";")

# Extraer las columnas 'area\_m2' y 'prices' como características (X)

X = data[['area\_m2', 'prices']]

# Codificar la columna 'district' utilizando one-hot encoding

district\_encoded = pd.get\_dummies(data['district'])

# Concatenar las características y las columnas codificadas

X\_encoded = pd.concat([X, district\_encoded], axis=1)

# Extraer la columna 'district' codificada como la variable objetivo (y)

y = district\_encoded

# Crear una instancia del modelo de Random Forest Regressor

reg = RandomForestRegressor()

# Entrenar el modelo utilizando los datos de las características y la variable objetivo

reg.fit(X\_encoded, y)

from sklearn import metrics

# Leer el archivo CSV y asignar los datos a la variable 'data'

data = pd.read\_csv(data\_url, sep=";")

# Extraer la columna 'prices' como la variable objetivo (y)

y = data['prices']

# Crear una instancia del modelo de Random Forest Regressor

reg = RandomForestRegressor()

# Entrenar el modelo utilizando los datos de la columna 'prices'

reg.fit(data['area\_m2'].values.reshape(-1, 1), y)

# Predecir los valores de la variable objetivo utilizando los datos de entrenamiento

y\_pred = reg.predict(data['area\_m2'].values.reshape(-1, 1))

# Calcular las métricas de evaluación

r2 = metrics.r2\_score(y, y\_pred)

adj\_r2 = 1 - (1 - r2) \* (len(y) - 1) / (len(y) - 1 - 1)

mae = metrics.mean\_absolute\_error(y, y\_pred)

mse = metrics.mean\_squared\_error(y, y\_pred)

rmse = np.sqrt(mse)

# Imprimir las métricas

print('R^2:', r2)

print('Adjusted R^2:', adj\_r2)

print('MAE:', mae)

print('MSE:', mse)

print('RMSE:', rmse)

R^2: 0.8099865079271028

Adjusted R^2: 0.8099282751682695

MAE: 154273.15765091797

MSE: 80816693231.68741

RMSE: 284282.7698466571

import matplotlib.pyplot as plt

plt.scatter(raw\_df['area\_m2'], raw\_df['prices'])

plt.xlabel('Area (m2)')

plt.ylabel('Prices')

plt.title('Scatter Plot of Area vs Prices')

plt.show()