# Importar las bibliotecas necesarias

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

# 'df' es tu DataFrame

# Seleccionar las características (X) y la variable objetivo (y)

X = df.drop('Price', axis=1)

y = df['Price']

# Dividir el conjunto de datos en entrenamiento (80%) y prueba (20%)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

Regresión lineal

# Crear y entrenar el modelo de regresión lineal

linear\_model = LinearRegression()

linear\_model.fit(X\_train, y\_train)

linear\_predictions = linear\_model.predict(X\_test)

# Calcular métricas de regresión lineal

linear\_rmse = np.sqrt(mean\_squared\_error(y\_test, linear\_predictions))

linear\_mse = mean\_squared\_error(y\_test, linear\_predictions)

linear\_mae = mean\_absolute\_error(y\_test, linear\_predictions)

linear\_r2 = r2\_score(y\_test, linear\_predictions)

# Imprimir métricas de regresión lineal

print(f'Regresión Lineal Metrics:')

print(f'RMSE: {linear\_rmse}')

print(f'MSE: {linear\_mse}')

print(f'MAE: {linear\_mae}')

print(f'R^2 Score: {linear\_r2}')

Arboles de decisión

# Crear y entrenar el modelo de árbol de decisión para regresión

tree\_model = DecisionTreeRegressor()

tree\_model.fit(X\_train, y\_train)

tree\_predictions = tree\_model.predict(X\_test)

# Calcular métricas de árbol de decisión

tree\_rmse = np.sqrt(mean\_squared\_error(y\_test, tree\_predictions))

tree\_mse = mean\_squared\_error(y\_test, tree\_predictions)

tree\_mae = mean\_absolute\_error(y\_test, tree\_predictions)

tree\_r2 = r2\_score(y\_test, tree\_predictions)

# Imprimir métricas de árbol de decisión

print(f'\nÁrbol de Decisión Metrics:')

print(f'RMSE: {tree\_rmse}')

print(f'MSE: {tree\_mse}')

print(f'MAE: {tree\_mae}')

print(f'R^2 Score: {tree\_r2}')

Bosques aleatorios

# Crear y entrenar el modelo de bosques aleatorios para regresión

forest\_model = RandomForestRegressor()

forest\_model.fit(X\_train, y\_train)

forest\_predictions = forest\_model.predict(X\_test)

# Calcular métricas de bosques aleatorios

forest\_rmse = np.sqrt(mean\_squared\_error(y\_test, forest\_predictions))

forest\_mse = mean\_squared\_error(y\_test, forest\_predictions)

forest\_mae = mean\_absolute\_error(y\_test, forest\_predictions)

forest\_r2 = r2\_score(y\_test, forest\_predictions)

# Imprimir métricas de bosques aleatorios

print(f'\nBosques Aleatorios Metrics:')

print(f'RMSE: {forest\_rmse}')

print(f'MSE: {forest\_mse}')

print(f'MAE: {forest\_mae}')

print(f'R^2 Score: {forest\_r2}')

KNN

# Crear y entrenar el modelo de KNN para regresión

knn\_model = KNeighborsRegressor()

knn\_model.fit(X\_train, y\_train)

knn\_predictions = knn\_model.predict(X\_test)

# Calcular métricas de KNN

knn\_rmse = np.sqrt(mean\_squared\_error(y\_test, knn\_predictions))

knn\_mse = mean\_squared\_error(y\_test, knn\_predictions)

knn\_mae = mean\_absolute\_error(y\_test, knn\_predictions)

knn\_r2 = r2\_score(y\_test, knn\_predictions)

# Imprimir métricas de KNN

print(f'\nKNN Metrics:')

print(f'RMSE: {knn\_rmse}')

print(f'MSE: {knn\_mse}')

print(f'MAE: {knn\_mae}')

print(f'R^2 Score: {knn\_r2}')

SVM

# Crear y entrenar el modelo de SVM para regresión

svm\_model = SVR()

svm\_model.fit(X\_train, y\_train)

svm\_predictions = svm\_model.predict(X\_test)

# Calcular métricas de SVM

svm\_rmse = np.sqrt(mean\_squared\_error(y\_test, svm\_predictions))

svm\_mse = mean\_squared\_error(y\_test, svm\_predictions)

svm\_mae = mean\_absolute\_error(y\_test, svm\_predictions)

svm\_r2 = r2\_score(y\_test, svm\_predictions)

# Imprimir métricas de SVM

print(f'\nSVM Metrics:')

print(f'RMSE: {svm\_rmse}')

print(f'MSE: {svm\_mse}')

print(f'MAE: {svm\_mae}')

print(f'R^2 Score: {svm\_r2}')