# Sử dụng thuật toán K-means để phân cụm dữ liệu

**Bước 1: Sử dụng các thư viện**

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import pandas as pd

import time

import seaborn as sns

from sklearn.metrics import silhouette\_score, davies\_bouldin\_score

**Bước 2: Đọc và chọn dữ liệu**

Đọc dữ liệu từ file Excel

#đọc dữ liệu

doc=pd.read\_csv("C:/Users/ACER/Desktop/STUDY/hoc\_may\_1/doanHocmay/air\_fixed.csv")

print(doc)

Chọn tất cả các biến cần quan tâm từ DataFrame gán bằng biến “features”

#chon du lieu

data = ['PM 2.5','temp','pressure','humidity','wind\_speed']

features = doc[data]

**Bước 3: Chọn giá trị k (cụm) phù hợp bằng phương pháp Elbow và trực quan hóa kết quả**

**Chọn số lượng cụm K:** Sử dụng phương pháp Elbow để chọn số lượng cụm phù hợp.

#chọn k phù hợp bằng elbow methods

# Chọn số lượng cụm K

k\_values = range(1, 11)

sse\_values = []

fit\_times = []

# Thực hiện K-means cho mỗi giá trị K và tính SSE và thời gian thực hiện

for k in k\_values:

    start\_time = time.time()

    kmeans = KMeans(n\_clusters=k, random\_state=42)

    kmeans.fit(features)

    fit\_time = time.time() - start\_time

    sse\_values.append(kmeans.inertia\_)  # inertia\_ chính là SSE

    fit\_times.append(fit\_time)

**Vẽ đồ thị Elbow và thời gian (fit time) thực hiện K-means:** Trực quan hóa để chọn K tối ưu.

# Vẽ đồ thị Elbow và đồ thị thời gian thực hiện K-means

fig, ax1 = plt.subplots()

# Đồ thị Elbow

color = 'tab:red'

ax1.set\_xlabel('Number of Clusters (K)')

ax1.set\_ylabel('Sum of Squared Errors (SSE)', color=color)

ax1.plot(k\_values, sse\_values, marker='o', color=color)

ax1.tick\_params(axis='y', labelcolor=color)

# Tạo đồ thị thời gian thực hiện K-means trên cùng trục x

ax2 = ax1.twinx()

color = 'tab:blue'

ax2.set\_ylabel('Fit Time (s)', color=color)

ax2.plot(k\_values, fit\_times, marker='s', color=color)

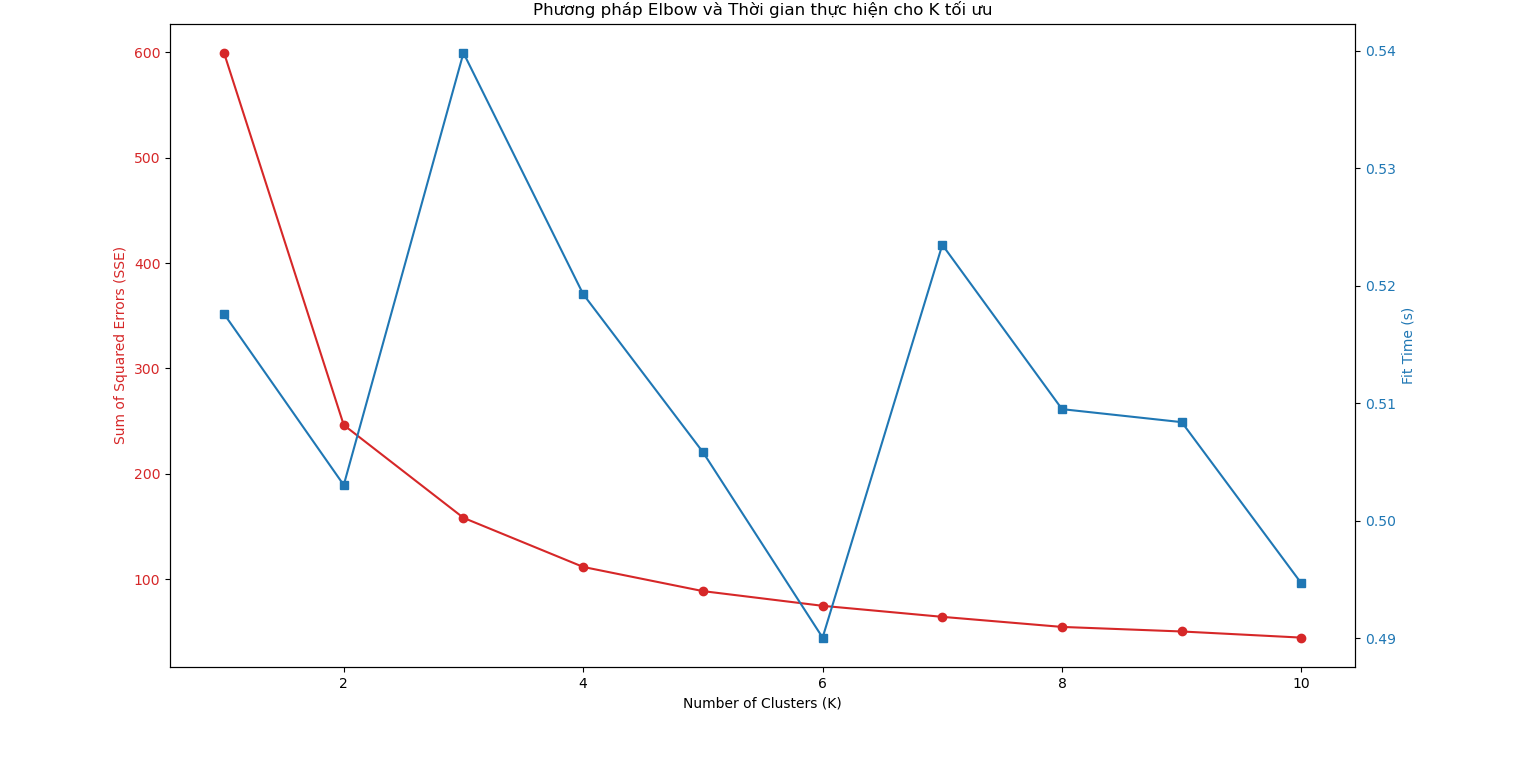
ax2.tick\_params(axis='y', labelcolor=color)

fig.tight\_layout()

plt.title('Phương pháp Elbow và Thời gian thực hiện cho K tối ưu')

plt.show()

Kết quả thu được:



Dựa trên kết quả thu được, chúng em xác định số cụm (k) bằng 2.

**Bước 4:** **Tính các giá trị Silhouette Score và Davies bouldin Index** để đánh giá chất lượng phân cụm

# Tính Silhouette Score

silhouette\_avg = silhouette\_score(features, kmeans.labels\_)

print(f"Silhouette Score: {silhouette\_avg}")

# Tính Davies-Bouldin Index

davies\_bouldin = davies\_bouldin\_score(features, kmeans.labels\_)

print(f"Davies-Bouldin Index: {davies\_bouldin}")

Kết quả: **Silhouette Score:** 0.53

**Davies-Bouldin Index:** 0.73

**Bước 5: Dùng K-means để phân cụm dữ liệu:** Áp dụng K-means với số cụm đã chọn

#du dung Kmeans de phan cum du lieu

kmeans = KMeans(n\_clusters=2, random\_state=42)

kmeans.fit(features)

# lay cac diem trong tam lam cum

centroids = kmeans.cluster\_centers\_

# Lấy nhãn của từng điểm dữ liệu

labels = kmeans.labels\_

**Hiển thị biểu đồ pairplot của các biến:** Thể hiện sự phân bố và tương quan giữa các biến trước khi phân cụm.

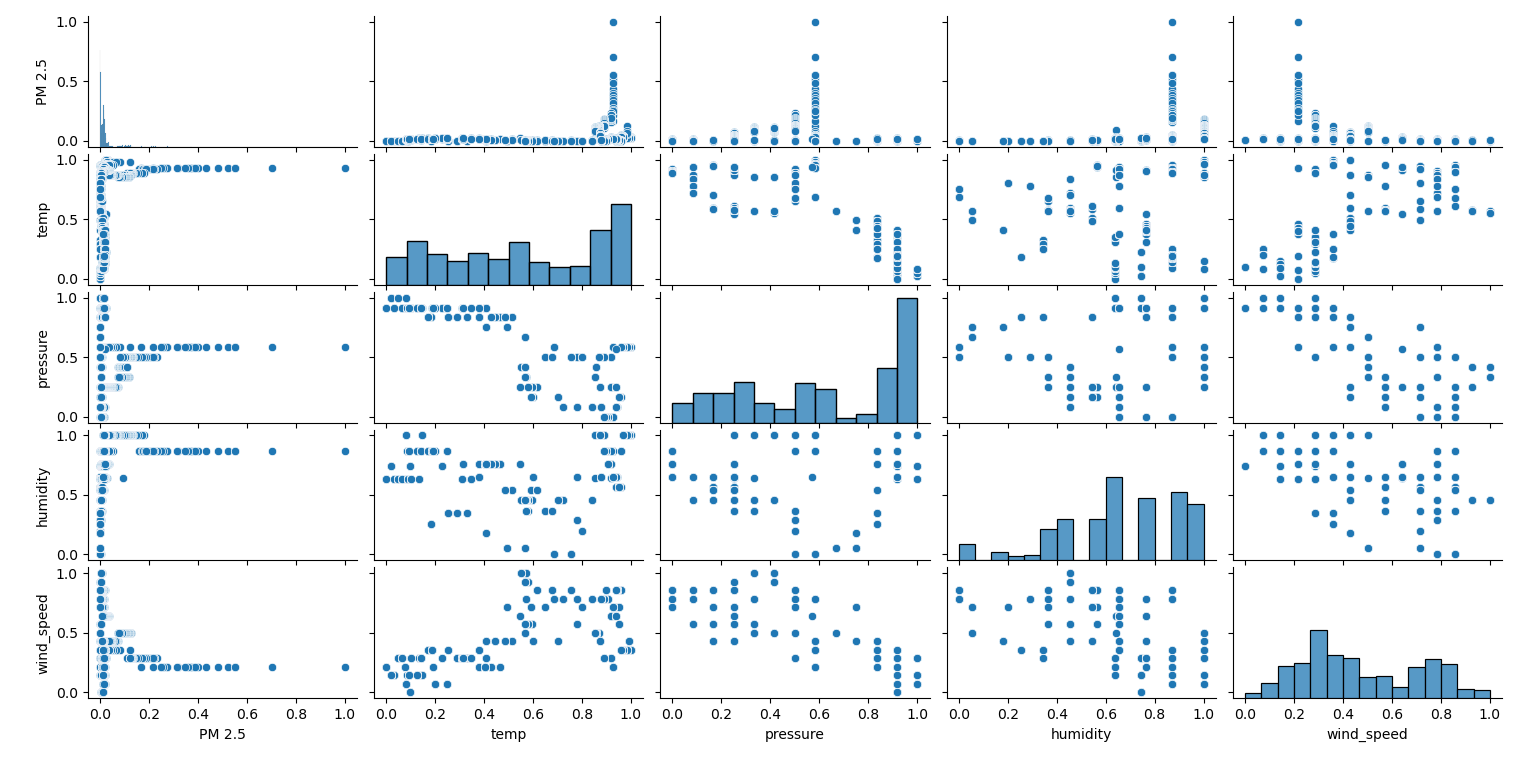
# Visualize pairplot để xem sự phân bố và tương quan giữa các biến

sns.pairplot(features)

# plt.title('Biểu đồ Pairplot của Các Biến Được Chọn')

plt.show()

Kết quả chúng em thu được:



Biểu đồ Pairplot của các biến trước khi sử dụng K-means để phân cụm

* Biểu đồ này hiển thị phân bố và tương quan giữa các biến 'PM 2.5', 'temp', 'pressure', 'humidity', 'wind\_speed' trước khi thực hiện phân cụm.
* Mỗi scatter plot ngoài đường chéo chính thể hiện mối quan hệ giữa cặp biến, trong khi histogram trên đường chéo chính thể hiện phân bố của từng biến.

**Hiển thị biểu đồ pairplot của các biến với màu sắc biểu thị cụm:** Thể hiện kết quả phân cụm của K-means trên biểu đồ pairplot.

# Thực hiện K-means và visualize kết quả

kmeans = KMeans(n\_clusters=2, random\_state=42)

kmeans.fit(features)

# Thêm nhãn cụm vào DataFrame

doc['Cluster'] = kmeans.labels\_

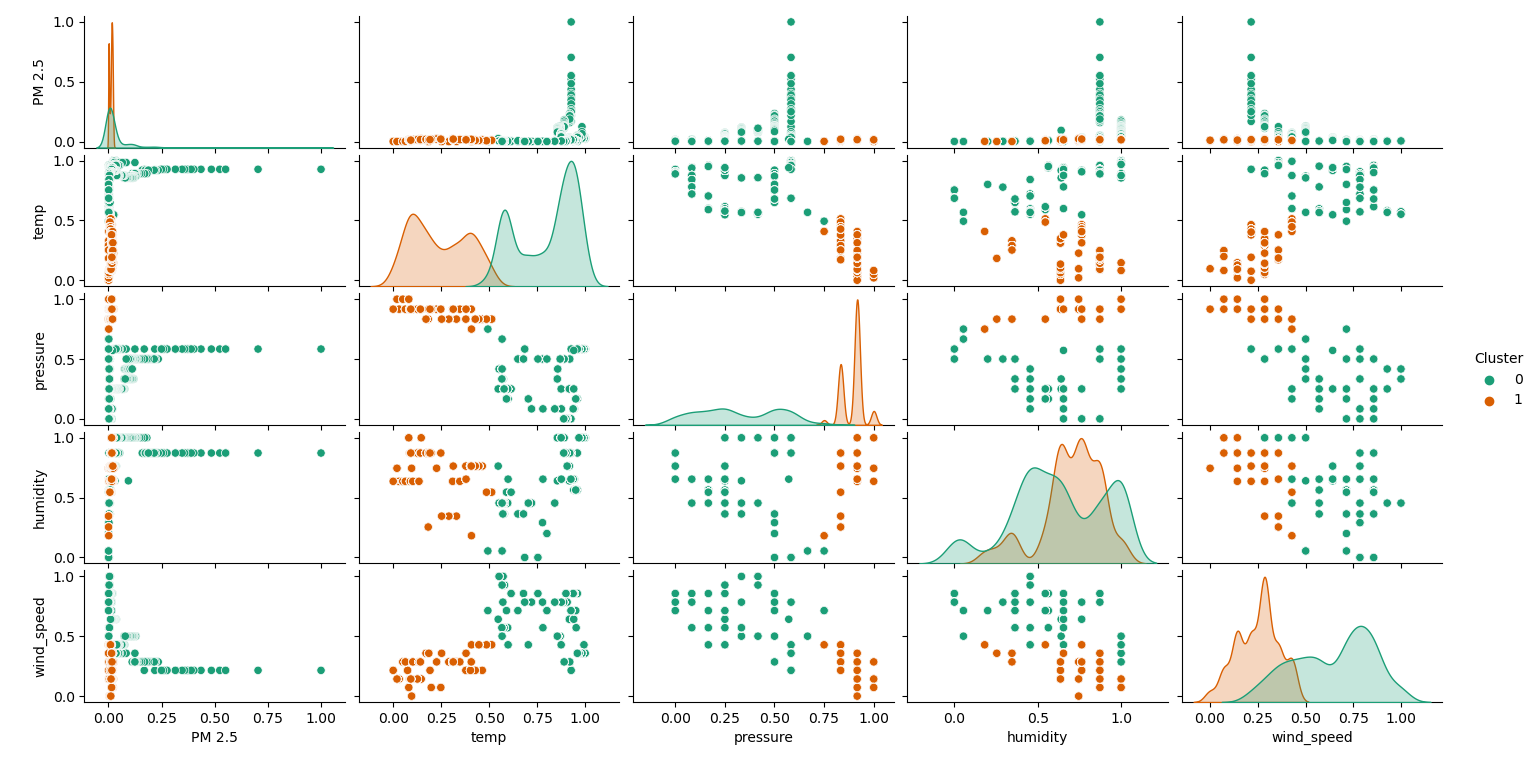
# Visualize pairplot của các biến được chọn với màu sắc được phân loại theo cụm

sns.pairplot(doc, hue='Cluster', palette='Dark2')

# plt.title('Biểu đồ Pairplot của Các Biến Được Chọn với Cụm')

plt.show()

Kết quả thu được:



Biểu đồ Pairplot của các biến sau khi sử dụng K-means để phân cụm:

* Biểu đồ này sử dụng màu sắc để biểu thị nhãn cụm mà K-means đã gán cho mỗi điểm dữ liệu.
* Các scatter plots ngoài đường chéo chính giúp xem xét mối quan hệ giữa các biến trong từng cụm.
* Mỗi cụm có thể có các đặc điểm riêng biệt, và sự tương đồng giữa các điểm trong cùng một cụm được thể hiện bằng màu sắc.

**Kết luận:**

# Sử dụng thuật toán K-nearest neighbors Classifier để phân loại giá nhà

**Bước 1:** Sử dụng các thư viện

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

**Bước 2:** Đọc dữ liệu từ file Excel

# Đọc dữ liệu từ file Excel

df=pd.read\_csv('C:/Users/ACER/Desktop/STUDY/hoc\_may\_1/doanHocmay/house\_price\_fixed.csv')

**Bước 3:** Tiền xử lí dữ liệu

Tiến hành chia xột ‘Price’ thành 3 mức 0, 1, 2 tương ứng với “Rẻ, Trung bình, Đắt’.

# Chia cột 'Price' thành 3 mức 0, 1, 2

price\_bins = [0, 250000, 350000, 450000]

price\_labels = [0, 1, 2]

df['Price\_Category'] = pd.cut(df['Price'], bins=price\_bins, labels=price\_labels)

**Bước 4:** Chọn đặc trưng và tiến hành chia dữ liệu thành tập huấn luyện và tập kiểm tra với 80% là tập huấn luyện và 20% là tập kiểm tra.

# Chọn các đặc trưng để sử dụng trong mô hình KNN

features = df[['SquareFeet', 'Bedrooms', 'Bathrooms', 'YearBuilt']]

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, df['Price\_Category'], test\_size=0.2, random\_state=42)

**Bước 5:** Chuẩn hóa dữ liệu

Chuẩn hóa dữ liệu để đảm bảo các đặc trưng có cùng thang đo, giúp mô hình học hiệu quả hơn.

# Chuẩn hóa dữ liệu

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

**Bước 6:** Xây dựng mô hình KNN

Sử dụng thuật toán K-Nearest Neighbors (KNN) để xây dựng mô hình phân loại, với số lượng láng giềng là 3.

# Xây dựng mô hình KNN

knn\_model = KNeighborsClassifier(n\_neighbors=3)

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

**Bước 7:** Dự đoán và đánh giá mô hình

Dự đoán trên tập kiểm tra và đánh giá mô hình sử dụng độ chính xác Accuracy và ma trận nhầm lẫn.

# Dự đoán trên tập kiểm tra

y\_pred = knn\_model.predict(X\_test\_scaled)

# Đánh giá mô hình

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print(f'Accuracy: {accuracy}')

print(f'Confusion Matrix:\n{conf\_matrix}')

Kết quả Accuracy: 0.69

Ma trận nhầm lẫn:

[[306 65 1]

[ 84 108 4]

[ 7 23 2]]

**Bước 8:** Trực quan hóa kết quả

Trực quan hóa ma trận nhầm lẫn bằng heatmap để thấy cách mô hình phân loại giá nhà vào các mức giá.

# Trực quan hóa kết quả

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=price\_labels, yticklabels=price\_labels)

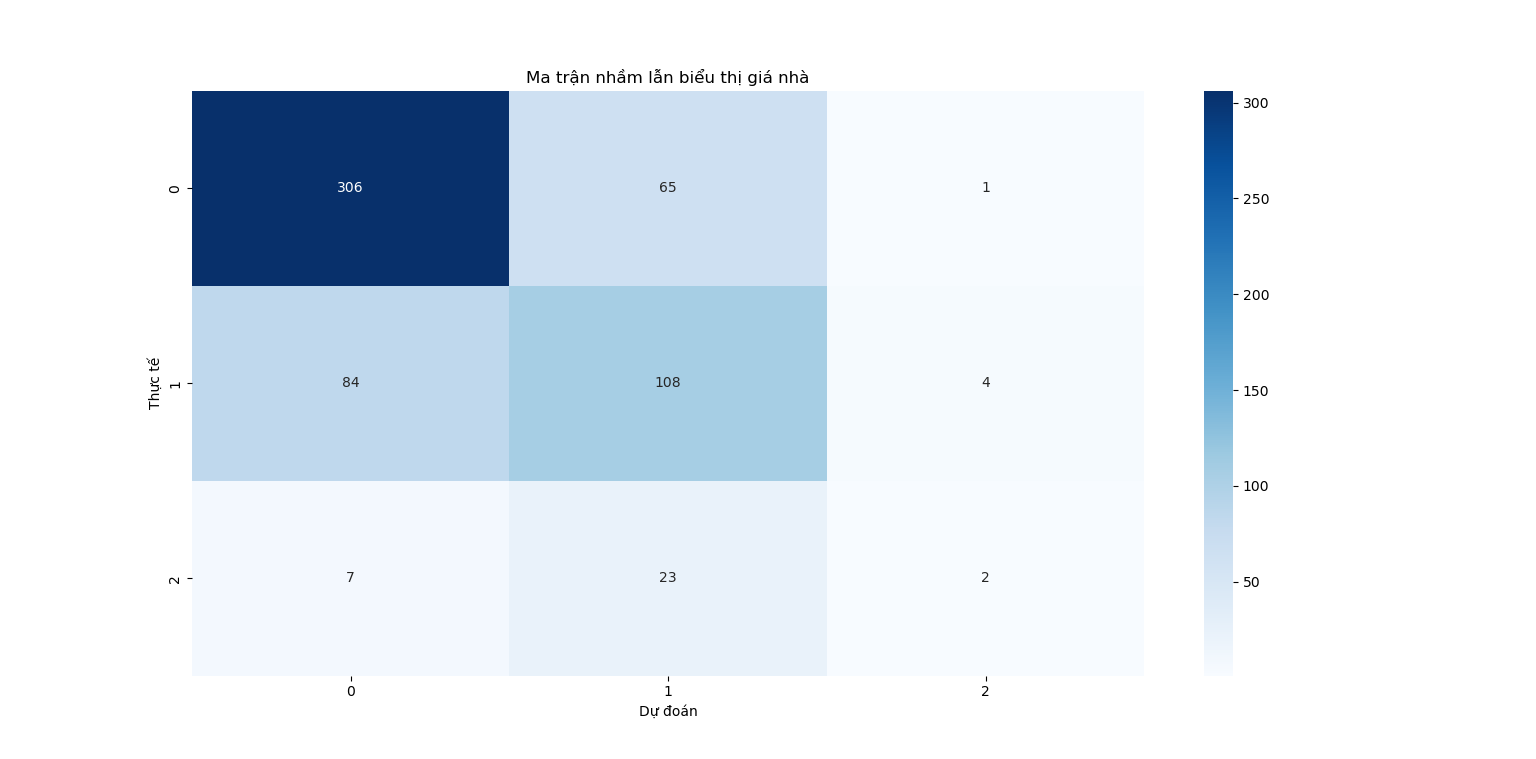
plt.xlabel('Dự đoán')

plt.ylabel('Thực tế')

plt.title('Ma trận nhầm lẫn biểu thị giá nhà')

plt.show()

Kết quả thu được:



**Kết luận:**

# Sử dụng thuật toán K-nearest neighbors Regressor để phân loại giá nhà

**Bước 1:** Sử dụng thư viện

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsRegressor

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

import matplotlib.pyplot as plt

**Bước 2:** Đọc dữ liệu từ file Excel

# Đọc dữ liệu từ file Excel

df=pd.read\_csv('C:/Users/ACER/Desktop/STUDY/hoc\_may\_1/doanHocmay/house\_price\_fixed.csv')

**Bước 3:** Chọn đặc trưng và biến mục tiêu

# Chọn các đặc trưng và biến mục tiêu

features = df[['SquareFeet', 'Bedrooms', 'Bathrooms', 'YearBuilt']]

target = df['Price']

**Bước 4:** Chia dữ liệu thành tập huấn luyện và kiểm thử

# Chia dữ liệu thành tập huấn luyện và tập kiểm thử

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

**Bước 5:** Chuẩn hóa dữ liệu để đảm bảo các đặc trưng có cùng tỷ lệ

# Chuẩn hóa dữ liệu

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

**Bước 6:** Xây dựng mô hình hồi quy K-nearest Neighbors Regressor

# Xây dựng mô hình hồi quy K-NN

knn\_model = KNeighborsRegressor(n\_neighbors=3)

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

**Bước 7:** Dự đoán trên tập kiểm thử và đánh giá mô hình

# Dự đoán trên tập kiểm thử

y\_pred = knn\_model.predict(X\_test\_scaled)

# Đánh giá mô hình

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'Mean Squared Error (MSE): {mse}')

print(f'R-squared (R2): {r2}')

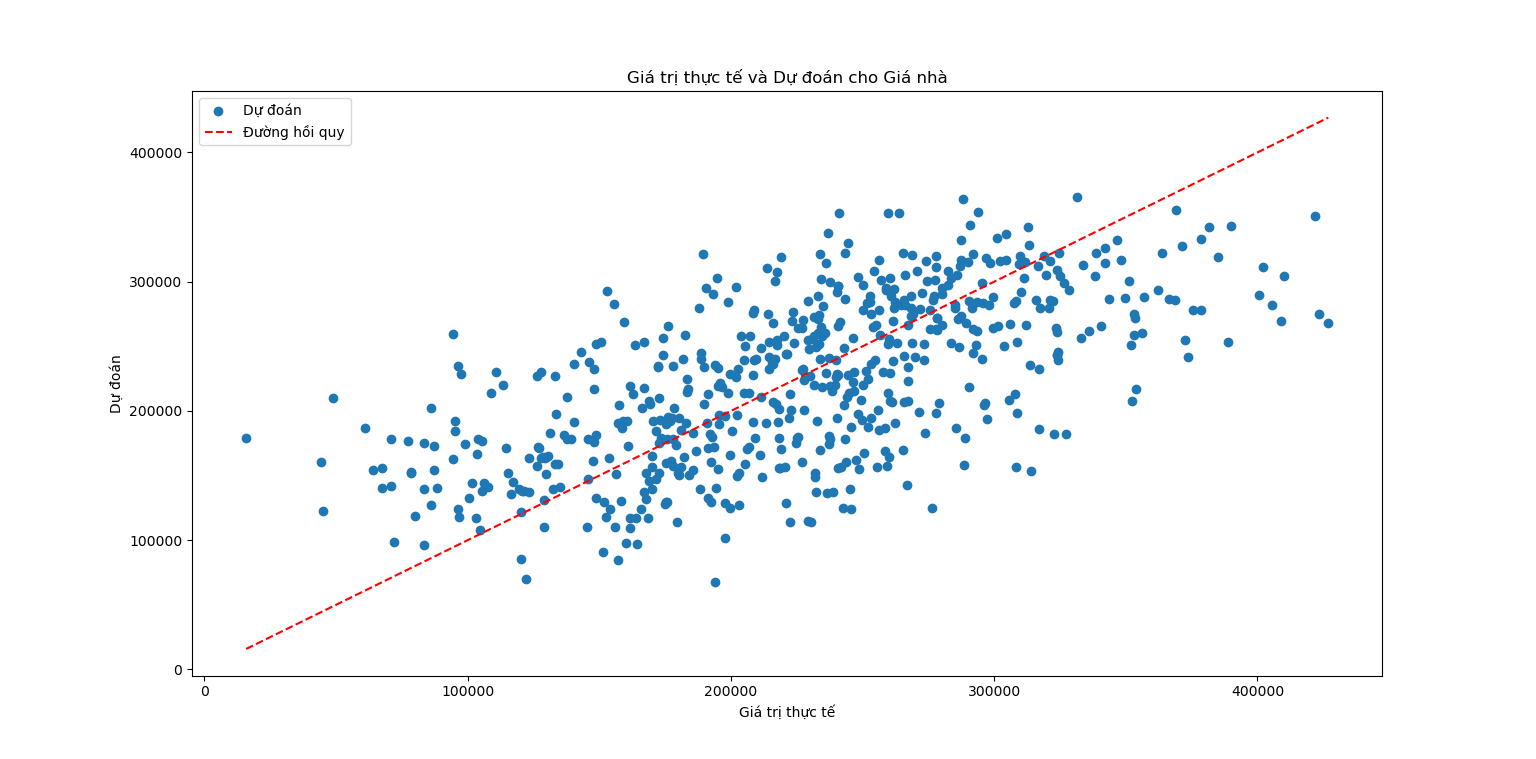
Kết quả: MSE= 3546796670.654247

R-squared (R2)=0.36

**Bước 8:** Trực quan hóa kết quả

Vẽ đồ thị phân tán giữa giá trị thực tế và giá trị dự đoán, và thêm đường hồi quy dự kiến (đường chéo đỏ). Điều này giúp hiển thị mức độ chính xác của mô hình và mối quan hệ giữa dự đoán và giá trị thực tế.

Kết quả:



**Kết luận:**

# Sử dụng hồi quy tuyến tính – Linear Regression để dự đoán giá nhà

**Bước 1:** Sử dụng thư viện

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import r2\_score

**Bước 2:** Đọc dữ liệu từ file Excel

# Đọc dữ liệu từ DataFrame

df=pd.read\_csv('C:/Users/ACER/Desktop/STUDY/hoc\_may\_1/doanHocmay/house\_price\_fixed.csv')

**Bước 3:** Chia dữ liệu thành tập huấn luyện và kiểm tra

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra

X = df[['SquareFeet', 'Bedrooms', 'Bathrooms', 'Neighborhood', 'YearBuilt']]

y = df['Price']

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

**Bước 4:** Chuẩn hóa dữ liệu

Sử dụng StandarScaler để chuẩn hóa dữ liệu để có giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1.

# Chuẩn hóa dữ liệu

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

**Bước 5:** Xây dựng mô hình Linear Regression

Sử dụng thư viện scikit-learn để xây dựng mô hình Linear Regression trên tập huấn luyện.

# Xây dựng mô hình Linear Regression

linear\_model = LinearRegression()

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

**Bước 6:** Dự đoán và đánh giá mô hình

Dự đoán giá nhà trên tập kiểm thử và tính toán MSE và R-squred để đánh giá hiệu suất của mô hình.

# Dự đoán giá nhà trên tập kiểm thử

y\_pred = linear\_model.predict(X\_test\_scaled)

# Đánh giá mô hình

mse = mean\_squared\_error(y\_test, y\_pred)

print(f'Mean Squared Error: {mse}')

# Tính giá trị R2

r2 = r2\_score(y\_test, y\_pred)

print(f'R-squared: {r2}')

Kết quả MSE: 2700520941.828473

R-squared: 0.51

**Bước 7:** Tính Residuals bằng cách lấy hiệu giữa giá trị thực tế và giá trị dự đoán.

# Tính residuals

residuals = y\_test - y\_pred

**Bước 8:** Trực quan hóa kết quả

Vẽ biểu đồ mô hình hồi quy và Residuals:

# Vẽ biểu đồ mô hình hồi quy và residuals

plt.figure(figsize=(12, 6))

# Biểu đồ mô hình hồi quy

plt.subplot(1, 2, 1)

sns.regplot(x=y\_pred, y=y\_test, scatter\_kws={'alpha':0.5})

plt.title('Mô hình hồi quy')

# Biểu đồ residuals

plt.subplot(1, 2, 2)

sns.residplot(x=y\_pred, y=residuals, scatter\_kws={'alpha':0.5})

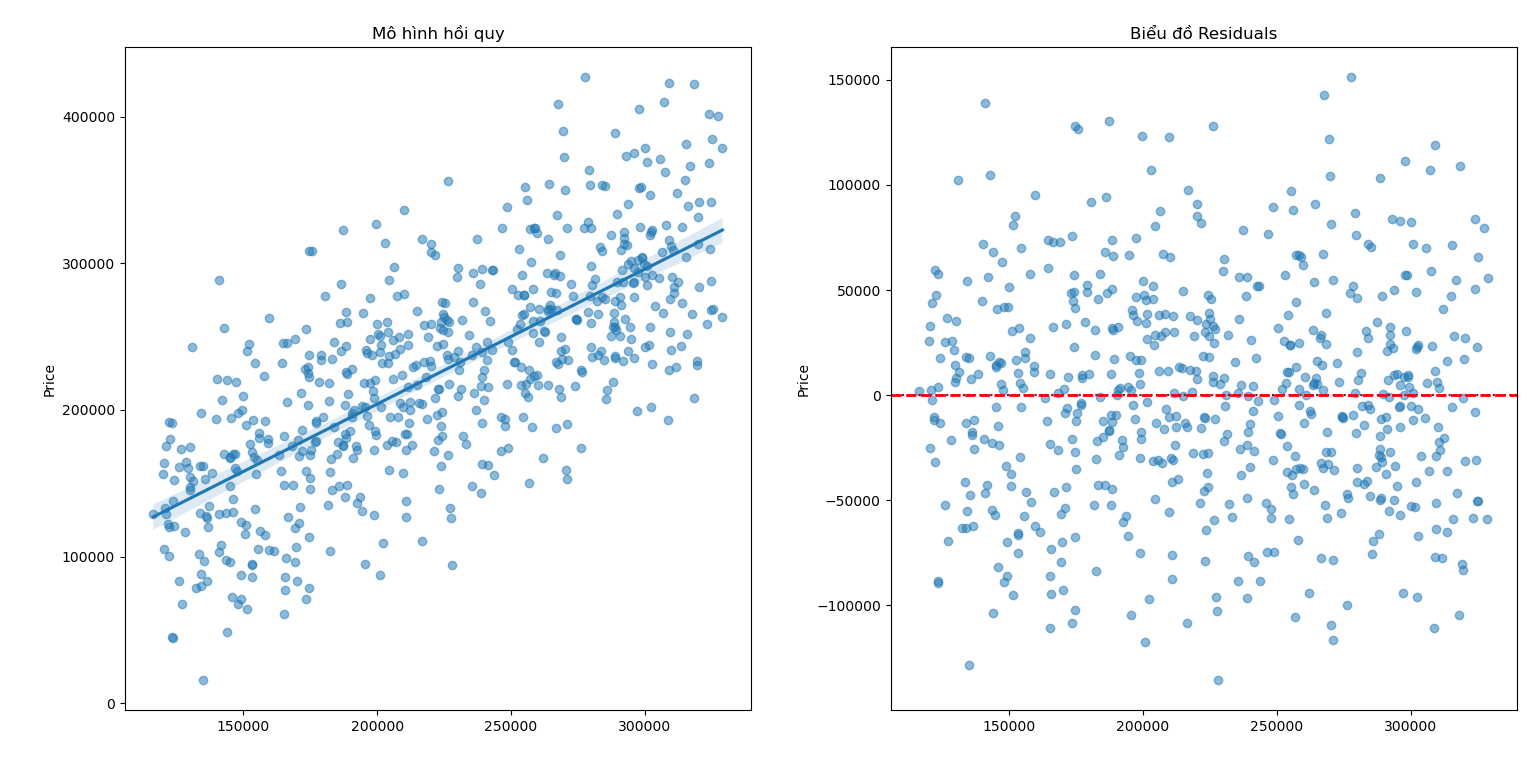
plt.axhline(y=0, color='red', linestyle='--', linewidth=2, label='Đường trung bình')

plt.title('Biểu đồ Residuals')

plt.tight\_layout()

plt.show()

Kết quả:



Biểu đồ mô hình hồi quy: Hiển thị scatter plot giữa giá trị dự đoán và giá trị thực tế.

Biểu đồ Residuals: Hiển thị scatter plot giữa giá trị dự đoán và residuals, cùng với đường trung bình.

**Kết luận:**