import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import plotly.graph\_objects as gb

import plotly.express as ex

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

from sklearn.linear\_model import LinearRegression

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

Data=pd.read\_csv("/kaggle/input/housing-price-prediction-data/housing\_price\_dataset.csv")

Data

fig=ex.scatter(data\_frame=Data,x="SquareFeet",y="Price",color="Neighborhood",size="YearBuilt",trendline="ols")

fig.show()

fig=ex.pie(Data,values="Price",names="Neighborhood")

fig.show()

Data["Neighborhood"].value\_counts()

Data["Neighborhood"]=Data["Neighborhood"].map({"Suburb":0,"Rural" :1,"Urban":2}).astype(int)

X=Data.iloc[:,:-1]

y=Data.iloc[:,-1]

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

lr=LinearRegression()

lr.fit(X\_train,y\_train)

# Make predictions on the training and testing data

y\_train\_pred = lr.predict(X\_train)

y\_test\_pred = lr.predict(X\_test)

# Calculate metrics for evaluation

train\_mse = mean\_squared\_error(y\_train, y\_train\_pred)

test\_mse = mean\_squared\_error(y\_test, y\_test\_pred)

train\_r2 = r2\_score(y\_train, y\_train\_pred)

test\_r2 = r2\_score(y\_test, y\_test\_pred)

# Create DataFrames for training and testing datasets

train\_data = pd.DataFrame({'Actual': y\_train.ravel(), 'Predicted': y\_train\_pred.ravel(), 'Dataset': 'Training'})

test\_data = pd.DataFrame({'Actual': y\_test.ravel(), 'Predicted': y\_test\_pred.ravel(), 'Dataset': 'Testing'})

# Concatenate both DataFrames

combined\_data = pd.concat([train\_data, test\_data], ignore\_index=True)

# Plot the results using Seaborn

sns.set(style='ticks')

plt.figure(figsize=(10, 6))

sns.scatterplot(x='Actual', y='Predicted', hue='Dataset', data=combined\_data, s=100, alpha=0.8)

plt.plot([min(y.min(), y\_train\_pred.min()) - 1, max(y.max(), y\_train\_pred.max()) + 1],

[min(y.min(), y\_train\_pred.min()) - 1, max(y.max(), y\_train\_pred.max()) + 1], 'k--')

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.title('Actual vs. Predicted (Training and Testing)')

plt.legend()

plt.grid(True)

plt.show()









