Task 06: Housing Prices Prediction with Polynomial Regression - Extended

Objective: In this assignment, you will explore and apply data preprocessing, feature scaling, train-test splitting, and polynomial regression to predict housing prices. You will evaluate the model's performance using RMSE, MAE, and the correlation coefficient.

Dataset: Use the Housing Prices dataset, containing housing data with the five features and a target label (price).

Instructions:

1. Import Libraries

```
# Import required libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from scipy.stats import pearsonr
```

2. Define functions for Feature Scaling, Polynomial Regression and Model Evaluation

a. Make a function to define scaling using MinMax, Standard and Robust scalers.

```
scaler = [MinMaxScaler(), StandardScaler(), RobustScaler()]
def scale(x,n):
    x_scaled = scaler[n].fit_transform(x)
    return x_scaled
```

b. Define function to apply Polynomial Regression with n degrees

```
# Function to perform polynomial regression and return predictions
def polynomial_regression(X_train, X_test, y_train, y_test, degree):
    # Generate polynomial features
    poly = PolynomialFeatures(degree=degree)
    X_train_poly = poly.fit_transform(X_train)
    X_test_poly = poly.transform(X_test)

    # Fit the model
    model = LinearRegression()
    model.fit(X_train_poly, y_train)

    # Make predictions
    y_pred = model.predict(X_test_poly)

    return y_pred
```

c. Define function to calculate evaluation metrics

```
# Function to calculate evaluation metrics
def evaluate_model(y_test, y_pred):
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    mae = mean_absolute_error(y_test, y_pred)
    corr_coef, _ = pearsonr(y_test, y_pred)
    return rmse, mae, corr_coef[0]
```

3. Perform training and testing of the model

a. Load the dataset

```
# Load the dataset
data = pd.read_csv('housing_price_dataset.csv')
data.head()
```

b. Convert 'Neighborhood' Feature to Numerical Values

- Convert the categorical `Neighborhood` feature into numerical values with the following mapping:
  - Urban = 3
  - Suburb = 2
  - Rural = 1

```
# Map 'Neighborhood' to numerical values
neighborhood_mapping = {'Urban': 3, 'Suburb': 2, 'Rural': 1}
data['Neighborhood'] = data['Neighborhood'].map(neighborhood_mapping)
```

c. Separate Features and Target Variable

- Define `X` as the features and `y` as the target variable.

```
# Separate features and target variable
X = data.drop(columns=['Price'])
y = data[['Price']]
```

d. Scale the features as well as the target.

```
# Scale the data [minmax - 0, standardscaler - 1, robustscaler - 2]
n = 1
```

```
X = scale(X_original,n)

y = scale(y_original,n)
```

e. Split the Data into Train and Test Sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

f. Evaluate models for degrees 1, 2, and 3

```
for degree in [1, 2, 3]:

    y_pred = polynomial_regression(X_train, X_test, y_train, y_test,degree)

    rmse, mae, corr_coef = evaluate_model(y_test, y_pred)

    print(f"Degree {degree}: RMSE={rmse:.2f}, MAE={mae:.2f}, Correlation
Coefficient={corr_coef:.2f}")
```

4. Form a report stating your methods, experimental setup, results and conclusions drawn.

Example evaluations:

Features	SquareFeet	Bedrooms	Bathrooms	SqurareFeet+Bathrooms+Bedrooms	All 5 features
RMSE↓					
MAE↓					
Corr↑					