**LAB PROGRAM 6: Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.datasets import fetch\_openml**

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

**from sklearn.preprocessing import StandardScaler**

**# Step 1: Load Boston Housing dataset**

**data = fetch\_openml(name='boston', version=1, as\_frame=True)**

**X = data.data[['RM']].values # Using 'RM' (Average rooms per dwelling) as feature**

**y = data.target.astype(float).values # Convert target to float**

**# Step 2: Split dataset into training and testing sets**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)**

**# Step 3: Normalize data for better performance**

**scaler = StandardScaler()**

**X\_train, X\_test = scaler.fit\_transform(X\_train), scaler.transform(X\_test) #calculates Mean and Std dev of X\_train**

**# Step 4: Define Locally Weighted Regression function**

**def locally\_weighted\_regression(X, y, x\_query, tau):**

**m = X.shape[0] #gets the number of training examples and X.shape[0] gives the number of rows**

**X\_bias = np.c\_[np.ones(m), X]** **# Add bias term**

**x\_query\_bias = np.r\_[1, x\_query]** **# Add bias term for query point**

**# Compute weights using Gaussian Kernel (Eq. 4.4)**

**W = np.diag(np.exp(-np.square(X - x\_query) / (2 \* tau\*\*2)).flatten())**

**# Solve weighted least squares: θ = (X'WX)^(-1) X'Wy**

**theta = np.linalg.pinv(X\_bias.T @ W @ X\_bias) @ (X\_bias.T @ W @ y)**

**return x\_query\_bias @ theta** **# Prediction**

**# Step 5: Compute LWR predictions**

**tau = 0.4 # Bandwidth parameter**

**y\_pred = np.array([locally\_weighted\_regression(X\_train, y\_train, x\_q, tau) for x\_q in X\_test])**

**# Step 6: Plot results**

**plt.figure(figsize=(14, 6))**

**# Subplot 1: Scatter plot for LWR results**

**plt.subplot(1, 2, 1)**

**plt.scatter(X\_train, y\_train, label='Training Data', color='blue', alpha=0.5)**

**plt.scatter(X\_test, y\_test, label='Test Data', color='green', alpha=0.5)**

**plt.scatter(X\_test, y\_pred, color='red', label='LWR Predictions', alpha=0.7)**

**plt.xlabel('Average Rooms per Dwelling (Normalized)')**

**plt.ylabel('House Price')**

**plt.title('Locally Weighted Regression on Boston Housing Data')**

**plt.legend()**

**# Subplot 2: Scatter plot for Predicted vs Actual Plot**

**plt.subplot(1, 2, 2)**

**plt.scatter(y\_test, y\_pred, color='blue', alpha=0.7)**

**plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], color='red', linestyle='--', label='Ideal Fit')**

**plt.xlabel('Actual House Price')**

**plt.ylabel('Predicted House Price')**

**plt.title('Prediction vs Actual Plot for LWR')**

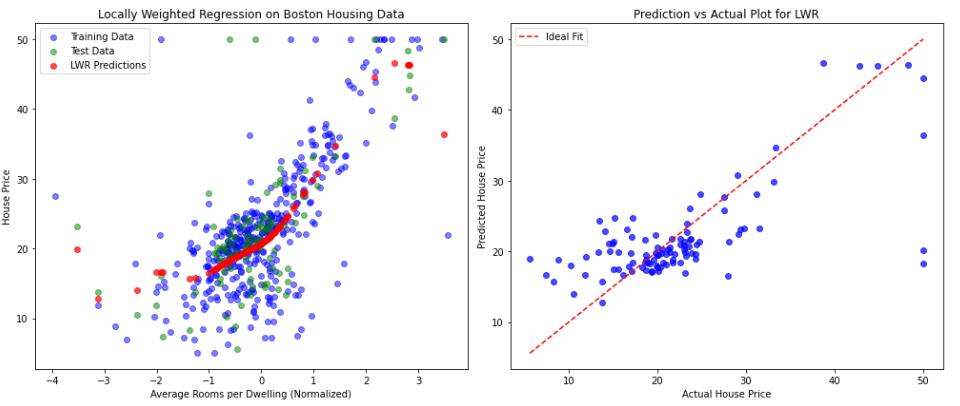
**plt.legend()**

**# Display both plots**

**plt.tight\_layout()**

**plt.show()**

**OUTPUT:**

****

**Boston Housing Dataset**

| **Feature** | **Description** |
| --- | --- |
| **CRIM** | Per capita crime rate by town |
| **ZN** | Proportion of residential and zoned for large lots |
| **INDUS** | Proportion of non-retail business acres |
| **CHAS** | Charles River dummy variable (1 if tract bounds river; 0 otherwise) |
| **NOX** | Nitric oxide concentration (pollution) |
| **RM** | **Average number of rooms per dwelling** |
| **AGE** | Proportion of owner-occupied units built before 1940 |
| **DIS** | Weighted distances to employment centers |
| **RAD** | Index of accessibility to radial highways |
| **TAX** | Property tax rate |
| **PTRATIO** | Pupil-teacher ratio |
| **B** | Measure of African American population (deprecated feature) |
| **LSTAT** | % of lower status population |
| **MEDV** | **Median value of homes** **(target variable)** |

**LAB PROGRAM 7: Develop a program to demonstrate the working of Linear Regression and Polynomial Regression. Use Boston Housing Dataset for Linear Regression and Auto MPG Dataset (for vehicle fuel efficiency prediction) for Polynomial Regression.**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns # visualization library built on top of matplotlib**

**import warnings # To control warning messages**

**from sklearn.datasets import fetch\_openml #loads datasets from openML-boston housing dataset**

**from sklearn.linear\_model import LinearRegression**

**from sklearn.preprocessing import PolynomialFeatures**

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

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

**warnings.filterwarnings("ignore")** **#Suppresses warning messages**

**# -----------------------------**

**# Part 1: Linear Regression on Boston Housing**

**# -----------------------------**

**# Load Boston Housing dataset via OpenML**

**boston = fetch\_openml(name="boston", version=1, as\_frame=True)**

**X\_boston = boston.data # all features**

**y\_boston = boston.target # Median house price**

**# Split dataset**

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

**# Train linear regression model**

**linear\_model = LinearRegression() #Initializes a model**

**linear\_model.fit(X\_train\_b, y\_train\_b) #Trains the model**

**y\_pred\_b = linear\_model.predict(X\_test\_b) #Model used to predict for the test set**

**# Evaluate**

**print("----- Linear Regression: Boston Housing -----")**

**print(f"MSE:{mean\_squared\_error(y\_test\_b, y\_pred\_b):.2f}")** **# Calculate Mean Squared Error- measures prediction error**

**print(f"RMSE: {np.sqrt(mean\_squared\_error(y\_test\_b, y\_pred\_b)):.2f}") # Calculate Root Mean Squared Error (RMSE)**

**print(f"R² Score: {r2\_score(y\_test\_b, y\_pred\_b):.2f}")** **# Calculate R-squared value- how well model explains variance91 is perfect)**

**print()**

**# -----------------------------**

**# Part 2: Polynomial Regression on Auto MPG**

**# -----------------------------**

**# Load Auto MPG dataset**

**df = sns.load\_dataset('mpg') # Loads the auto MPG dataset using seaborn**

**df = df.dropna() # drops rows with missing values to prevent errors**

**df['horsepower'] = df['horsepower'].fillna(df['horsepower'].median())** **# hp-measure of an engine’s power**

**X\_mpg = df[['horsepower']] # Selects horsepower as the feature**

**y\_mpg = df['mpg'] # Selects Miles Per Gallon as the target**

**# Split dataset**

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

**# Polynomial regression (degree 2)**

**poly = PolynomialFeatures(degree=2) #Quadratic equation**

**X\_train\_poly = poly.fit\_transform(X\_train\_m) #Transforms training and test input into polynomial form**

**X\_test\_poly = poly.transform(X\_test\_m)**

**poly\_model = LinearRegression()**

**poly\_model.fit(X\_train\_poly, y\_train\_m)**

**y\_pred\_poly = poly\_model.predict(X\_test\_poly)**

**# Evaluate**

**print("----- Polynomial Regression: Auto MPG (Degree 2) -----")**

**print(f"MSE:{mean\_squared\_error(y\_test\_m, y\_pred\_poly):.2f}")** **#Calculate Mean Squared Error**

**print(f"RMSE: {np.sqrt(mean\_squared\_error(y\_test\_m, y\_pred\_poly)):.2f}")** **# Calculate Root Mean Squared Error (RMSE)**

**print(f"R² Score: {r2\_score(y\_test\_m, y\_pred\_poly):.2f}")** **# Calculate R-squared value**

**print()**

**# Plot Polynomial Regression results**

**X\_range = np.linspace(X\_mpg.min(), X\_mpg.max(), 100).reshape(-1, 1) #creates 100 evenly spaced hp values within the range of the dataset and reshaped into 2D array to match model input format**

**X\_range\_poly = poly.transform(X\_range) #Transforms into polynomial forms**

**y\_range\_pred = poly\_model.predict(X\_range\_poly) #Predicts MPG for this range**

**plt.scatter(X\_mpg, y\_mpg, color='green', label='Actual Data')**

**plt.plot(X\_range, y\_range\_pred, color='red', label='Polynomial Fit (Degree 2)', linewidth=2)**

**plt.xlabel("Horsepower")**

**plt.ylabel("MPG")**

**plt.title("Polynomial Regression: Horsepower vs MPG")**

**plt.legend()**

**plt.grid(True)**

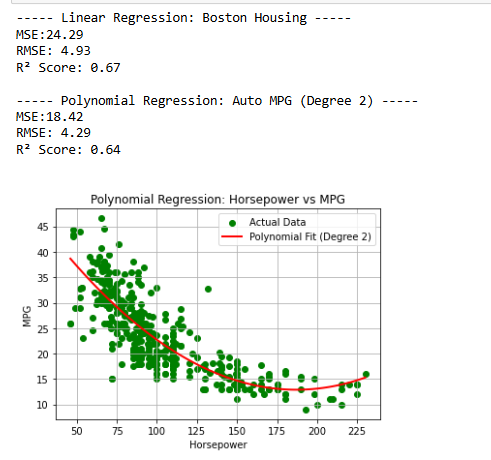
**plt.show()**

### ****Auto MPG Dataset:****

The **Auto MPG (miles per gallon)** dataset was published by the UCI Machine Learning Repository. It's used to **predict fuel efficiency (MPG)** of cars based on various characteristics like weight, engine size, and horsepower.

| **Feature** | **Description** |
| --- | --- |
| **mpg** | **Target variable — miles per gallon (fuel efficiency)** |
| **cylinders** | Number of cylinders (e.g., 4, 6, 8) |
| **displacement** | Engine displacement (cubic inches) |
| **horsepower** | **Engine power** |
| **weight** | Vehicle weight |
| **acceleration** | Time to accelerate from 0 to 60 mph (in seconds) |
| **model year** | Year of the car (e.g., 70 = 1970) |
| **origin** | Region of origin (1 = USA, 2 = Europe, 3 = Asia) |
| **car name** | Name of the car (e.g., "chevrolet impala") |

**OUTPUT:**



### ****1. Linear Regression: Boston Housing****

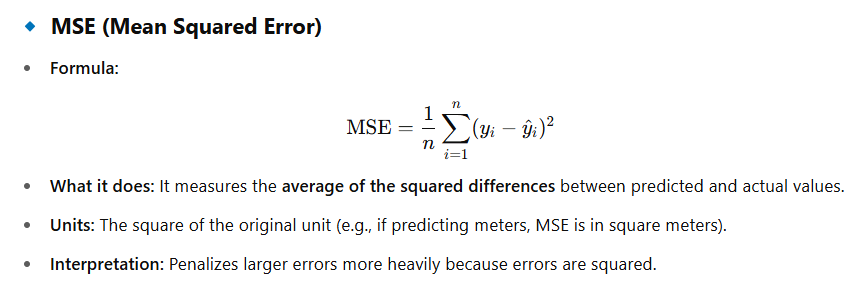
* **Dataset:** Boston Housing (predicting house prices based on features like crime rate, number of rooms, etc.)
* **MSE:** 24.29
  + The average squared difference between predicted and actual prices is 24.29 units².
* **RMSE:** 4.93
  + On average, the model’s predictions are off by about 4.93 units.
* **R² Score:** 0.67
  + The model explains **67% of the variance** in house prices, which is a decent fit.

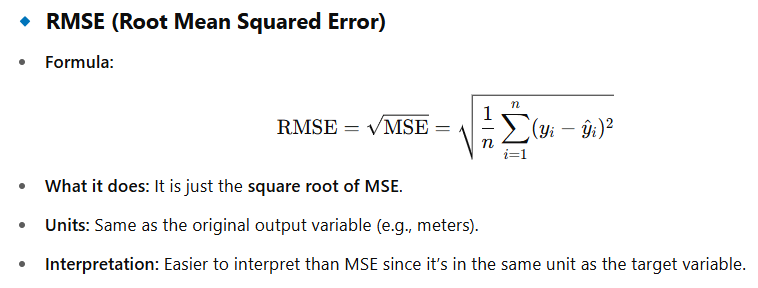
### ****2. Polynomial Regression (Degree 2): Auto MPG****

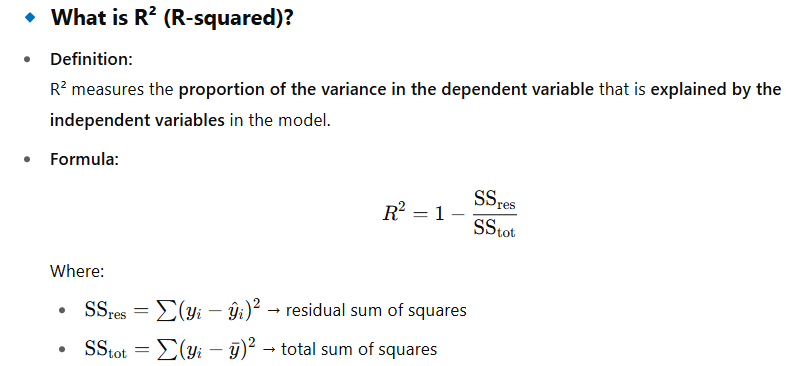
* **Dataset:** Auto MPG (predicting fuel efficiency or MPG based on features like horsepower)
* **MSE:** 18.42
  + Lower than the Boston model, indicating smaller squared errors.
* **RMSE:** 4.29
  + Slightly better average prediction error than the linear model.
* **R² Score:** 0.64
  + Explains **64% of the variance** in MPG — slightly lower than the Boston model despite lower error values.

**Graph:**

* There’s a **non-linear inverse relationship** — as horsepower increases, MPG generally decreases.
* The quadratic curve fits this relationship better than a straight line would.
* The red curve flattens at high horsepower, capturing that MPG levels off at higher power levels.

****

****

****