9	Aim	Understand and analyse the concept of Regression algorithm techniques.
	Program	Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

CONCEPT - Locally Weighted Regression (LWR)

Introduction

- Locally Weighted Regression (LWR), also known as Locally Weighted Scatterplot Smoothing (LOWESS), is a non-parametric regression method that fits multiple regressions in local neighbourhoods. It is especially useful for non-linear datasets.
- Non-parametric regression is a type of regression analysis that does not assume a predetermined form for the relationship between the independent and dependent variables. Instead, it seeks to model the relationship directly from the data, allowing for greater flexibility in capturing the underlying patterns and structures.

Key Concepts of LWR

1. **Local Fitting:**
   - For each query point x_q, LWR fits a local regression model using data points that are close to x_q.
   - The "closeness" is determined by a weighting function, typically a kernel function.
2. **Weighting Function:**
   - A kernel function K assigns weights to data points based on their distance from the query point.
   - A common choice is the Gaussian kernel:

$$K(x, x_i) = \exp\left(-\frac{(x - x_i)^2}{2\tau^2}\right)$$

3. **Weighted Linear Regression:**
   - For a given query point x_q, LWR solves a weighted linear regression problem:

$$\theta = (X^T W X)^{-1} X^T W y$$

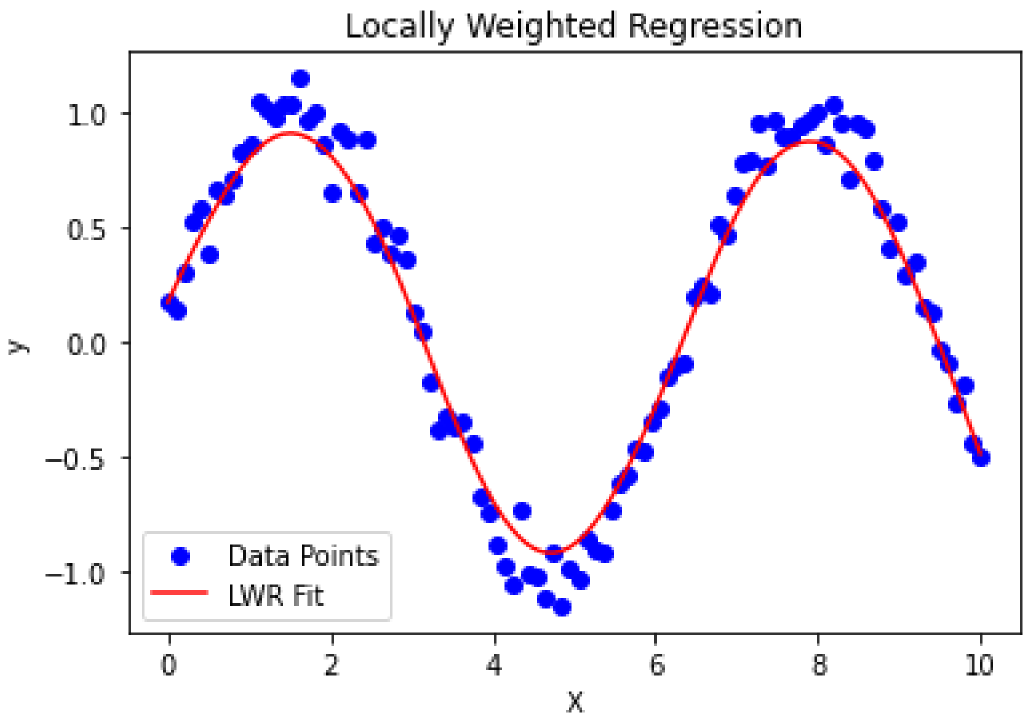
   where W is a diagonal matrix of weights calculated using the kernel function for each data point relative to x_q.

4. **Bandwidth Parameter (τ):**
   - The bandwidth parameter τ determines the scale of the neighborhood. A smaller τ results in a more localized model, while a larger τ leads to a smoother, more global model.

Program:

```
import numpy as np
import matplotlib.pyplot as plt

def kernel(x, x_i, tau):
    return np.exp(-np.sum((x - x_i)**2) / (2 * tau**2))

def locally_weighted_regression(X, y, tau):
    m = X.shape[0]
    y_pred = np.zeros(m)

    for i in range(m):
        weights = np.array([kernel(X[i], X[j], tau) for j in range(m)])
        W = np.diag(weights)
        XTX = X.T @ W @ X
        XTX_inv = np.linalg.pinv(XTX)
        theta = XTX_inv @ X.T @ W @ y
        y_pred[i] = X[i] @ theta

    return y_pred

# Generate synthetic dataset
np.random.seed(0)
X = np.linspace(0, 10, 100)
y = np.sin(X) + np.random.normal(scale=0.1, size=X.shape)

# Add a column of ones to include the intercept term
X_bias = np.c_[np.ones(X.shape[0]), X]

# Perform Locally Weighted Regression
tau = 0.5  # Bandwidth parameter
y_pred = locally_weighted_regression(X_bias, y, tau)

# Plot the results
plt.scatter(X, y, color='blue', label='Data Points')
plt.plot(X, y_pred, color='red', label='LWR Fit')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Locally Weighted Regression')
plt.legend()
plt.show()
```

Output:


Locally Weighted Regression
1.0
0.5
0.0
−0.5
−1.0
y
0
2
4
6
8
10
X
Data Points
LWR Fit