```
import numpy as np
from bokeh.plotting import figure, show
from bokeh.layouts import gridplot

# Radial kernel function
def radial_kernel(x0, X, tau):
    return np.exp(np.sum((X - x0) ** 2, axis=1) / (-2 * tau * tau))

# Local regression (Locally Weighted Regression)
def local_regression(x0, X, Y, tau):
    x0 = np.r_[1, x0]  # add bias term
    X_ = np.c_[np.ones(len(X)), X]  # design matrix with bias
    weights = radial_kernel(x0, X_, tau)
    xw = X_.T * weights  # weight each row
    beta = np.linalg.pinv(xw @ X_) @ xw @ Y
    return x0 @ beta

# Generate data
n = 1000
X = np.linspace(-3, 3, num=n)
print("The Data Set X (By Danish Shaikh 35):\n", X[1:10])

Y = np.log(np.abs(X ** 2 - 1) + 0.5)
print("\nThe Fitting Curve Data Set Y (By Danish Shaikh 35):\n", Y[1:10])

X += np.random.normal(scale=0.1, size=n)  # add noise
print("\nNormalized X (By Danish Shaikh 35):\n", X[1:10])

domain = np.linspace(-3, 3, num=300)
print("\nXo Domain Space (By Danish Shaikh 35):\n", domain[1:10])

# Plotting function
def plot_lwr(tau):
    prediction = [local_regression(x0, X, Y, tau) for x0 in domain]
    plot = figure(width=400, height=400)
    plot.title.text = f'tau={tau}'
    plot.scatter(X, Y, alpha=0.3)
    plot.line(domain, prediction, line_width=2, color='red')
    return plot

# Show results
show(gridplot([
    [plot_lwr(10.0), plot_lwr(1.0)],
    [plot_lwr(0.1), plot_lwr(0.01)]
]))
```

```
The Data Set X (By Danish Shaikh 35):
 [-2.99399399 -2.98798799 -2.98198198 -2.97597598 -2.96996997 -2.96396396
 -2.95795796 -2.95195195 -2.94594595]

The Fitting Curve Data Set Y (By Danish Shaikh 35):
 [2.13582188 2.13156806 2.12730467 2.12303166 2.11874898 2.11445659
 2.11015444 2.10584249 2.10152068]

Normalized X (By Danish Shaikh 35):
 [-3.22393056 -2.78687756 -2.99979968 -2.97780435 -2.99730802 -2.98447296
 -3.06460875 -2.96826371 -2.74559877]

Xo Domain Space (By Danish Shaikh 35):
 [-2.97993311 -2.95986622 -2.93979933 -2.91973244 -2.89966555 -2.87959866
 -2.85953177 -2.83946488 -2.81939799]
```

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

# Load dataset
data = pd.read_csv("tips.csv")

X_data = np.array(data.total_bill).reshape(-1, 1)
Y_data = np.array(data.tip).reshape(-1, 1)

# Add bias column
X = np.hstack((np.ones_like(X_data), X_data))
Y = Y_data
```

```
# Gaussian kernel (weights)
def gaussian_kernel(x0, X, tau):
    m = X.shape[0]
    W = np.eye(m)
    for i in range(m):
        diff = x0 - X[i]
        W[i, i] = np.exp(- (diff @ diff.T) / (2 * tau ** 2))
    return W

# Locally Weighted Linear Regression prediction
def lwlr_predict(x0, X, Y, tau):
    W = gaussian_kernel(x0, X, tau)
    theta = np.linalg.pinv(X.T @ W @ X) @ (X.T @ W @ Y)
    return x0 @ theta

# Fit over a domain of x-values
def lwlr_fit_domain(X, Y, tau, x_domain):
    y_pred = []
    for x0 in x_domain:
        x0_bias = np.array([1, x0]).reshape(1, 2)
        y_pred.append(lwlr_predict(x0_bias, X, Y, tau)[0, 0])
    return np.array(y_pred)

# Create domain
x_min, x_max = X_data.min(), X_data.max()
x_domain = np.linspace(x_min, x_max, 300)

# Try multiple tau values
taus = [0.1, 1, 10]

plt.figure(figsize=(8, 12))
for i, tau in enumerate(taus):
    plt.subplot(len(taus), 1, i+1)
    y_pred_domain = lwlr_fit_domain(X, Y, tau, x_domain)
    plt.scatter(X_data, Y, color='blue', alpha=0.5, label='Data')
    plt.plot(x_domain, y_pred_domain, color='red', linewidth=2, label=f'LWLR tau={tau}')
    plt.xlabel('Total Bill')
    plt.ylabel('Tip')
    plt.title(f'By Danish Shaikh 221P054 - LWLR Fit with tau={tau}')
    plt.grid(True)
    plt.legend()

plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-302723473.py:21: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is deprecated, and will error i
  W[i, i] = np.exp(- (diff @ diff.T) / (2 * tau ** 2))
```