```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from scipy.spatial.distance import cdist

df_linear =pd.read_csv("linear_dataset.csv")
df_lwr=pd.read_csv("lwr_dataset.csv")
df_poly=pd.read_csv("polynomial_dataset.csv")

df_linear.head()
```

	X	Y
0	0.00000	0.188315
1	0.10101	-0.033937
2	0.20202	0.073600
3	0.30303	0.395353
4	0.40404	0.275824

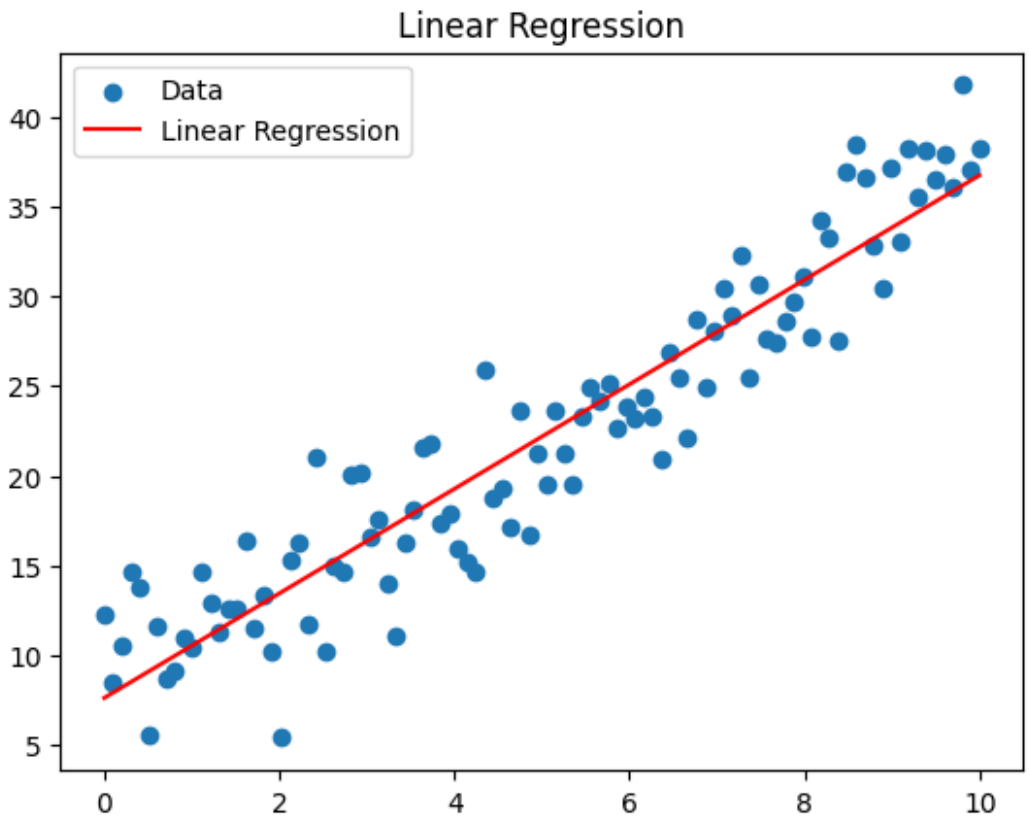
```
df_lwr.head()
```

	X	Y
0	0.00000	0.188315
1	0.10101	-0.033937
2	0.20202	0.073600
3	0.30303	0.395353
4	0.40404	0.275824

```
df_poly.head()
```

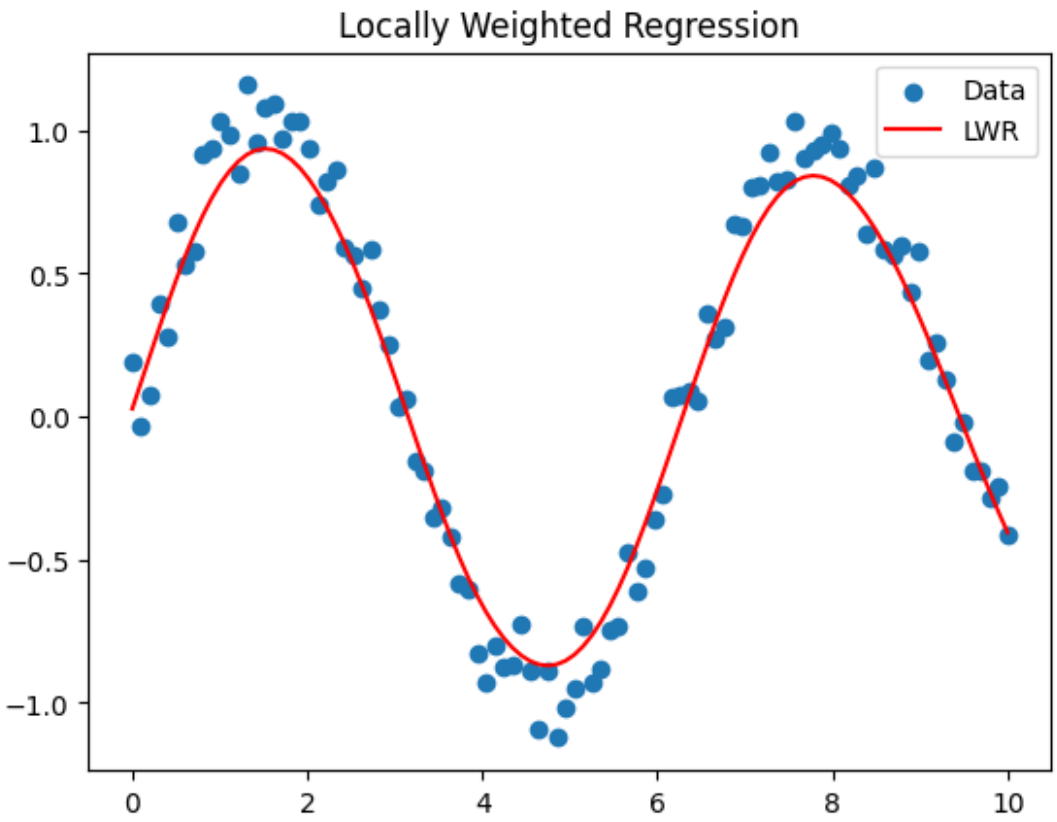
	X	Y
0	-5.00000	-202.359477
1	-4.89899	-204.474083
2	-4.79798	-187.545463
3	-4.69697	-164.156446
4	-4.59596	-157.491498

```
def linear_regression(df):
    X,y=df[['X']],df['Y']
    model=LinearRegression()
    model.fit(X,y)
    y_pred=model.predict(X)
    plt.scatter(X,y,label='Data')
    plt.plot(X,y_pred,color='red',label='Linear Regression')
    plt.legend()
    plt.title("Linear Regression")
    plt.show()
linear_regression(df_linear)
```

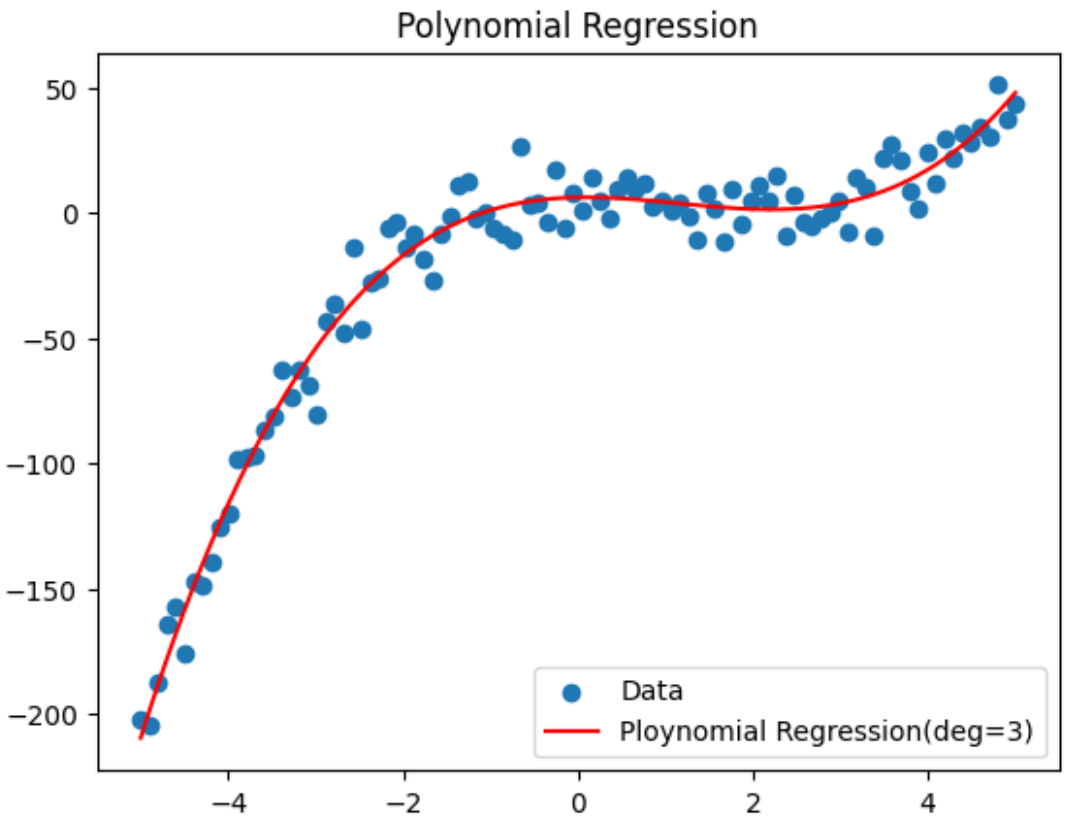



```
def gaussian_kernel(x,X,tau):
    return np.exp(-cdist([[x]],X,'sqeuclidean')/(2*tau**2))

def locally_weighted_regression(X_train,y_train,tau=0.5):
    X_train=np.hstack([np.ones((X_train.shape[0],1)),X_train])
    X_range=np.linspace(X_train[:,1].min(),X_train[:,1].max(),100)
    y_pred=[]
    for x in X_range:
        x_vec=np.array([1,x])
        weights=gaussian_kernel(x,X_train[:,1:],tau).flatten()
        W=np.diag(weights)
        theta=np.linalg.pinv(X_train.T @ W @ X_train)@(X_train.T @ W @
y_train)
        y_pred.append(x_vec @ theta)
    plt.scatter(X_train[:,1],y_train,label='Data')
    plt.plot(X_range,y_pred,color='red',label='LWR')
    plt.legend()
    plt.title("Locally Weighted Regression")
    plt.show()
locally_weighted_regression(df_lwr[['X']].values,df_lwr['Y'].values)
```




```
def polynomial_regression(df,degree=3):
    X,y=df[['X']],df['Y']
    model=make_pipeline(PolynomialFeatures(degree),LinearRegression())
    model.fit(X,y)
    y_pred=model.predict(X)
    plt.scatter(X,y,label='Data')
    plt.plot(X,y_pred,color='red',label=f'Ploynomial
Regression(deg={degree})')
    plt.legend()
    plt.title("Polynomial Regression")
    plt.show()
polynomial_regression(df_poly,degree=3)
```


Polynomial Regression
50
0
−50
−100
−150
−200
−4
−2
0
2
4
Data
Ploynomial Regression(deg=3)