# INTELLIGENT SYSTEMS LAB-7 (11/10/2021)

*__NAME__: Harshvardhan Agarwal | __SEC__: A | __REG NO__.: 201800524*

## PROBLEM STATEMENT

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

# Lab Assignment:1. Repeat the same experiment for different point (.5 to .05) and tau (10 to 100)

# Lab Assignment: 2 Perform LOESS or LOWESS using the dataset given below

1. Submit pdf with code and graph.
2. Perform the experiment for at least two data sets.
3. Analyze with different test cases.

## PROBLEM SOLUTION

## SOURCE CODE

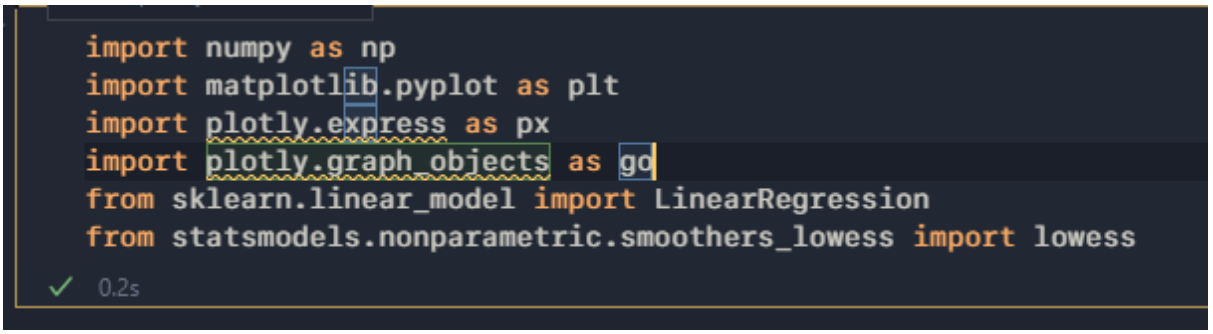
```
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go

from sklearn.linear_model import LinearRegression
from statsmodels.nonparametric.smoothers_lowess import lowess

# ASSIGNMENT 1

def wm(point, X, tau):
    m = X.shape[0]
    w = np.mat(np.eye(m))
    for i in range(m):
        xi = X[i]
        d = (-2 * tau * tau)
        w[i, i] = np.exp(np.dot((xi-point), (xi-point).T)/d)
    return w

def predict(X, y, point, tau):
    m = X.shape[0]
    X_ = np.append(X, np.ones(m).reshape(m,1), axis=1)
    point_ = np.array([point, 1])
```

```
    w = wm(point_, X_, tau)
    theta = np.linalg.pinv(X_.T*(w * X_))*(X_.T*(w * y))
    pred = np.dot(point_, theta)
    return theta, pred

def plot_predictions(X, y, tau, nval):
    X_test = np.linspace(-3, 3, nval)
    preds = []
    for point in X_test:
        theta, pred = predict(X, y, point, tau)
        preds.append(pred)
    X_test = np.array(X_test).reshape(nval,1)
    preds = np.array(preds).reshape(nval,1)
    plt.plot(X, y, 'b.')
    plt.plot(X_test, preds, 'r.')
    plt.show()

X = np.random.randn(1000,1)
y = 2 * (X ** 3) + 10 + 4.6 * np.random.randn(1000, 1)
plt.plot(X, y, 'b.')
plt.show()

for i in [0.5, 0.1, 0.05]:
    for j in [10, 50, 100]:
        print("\n\nTau: {}\nPoints: {}".format(i, j))
        plot_predictions(X, y, i, j)

# ASSIGNMENT 2

X = np.linspace(-3, 3, 1000)
print(X.shape)
X += np.random.normal(scale = 0.05, size = 1000)
Y = np.log(np.abs((X**2) - 1) + 0.5)
print(Y.shape)
plt.scatter(X, Y, alpha = 0.32)
plt.show()

X_Reshape = X.reshape(-1, 1)
model = LinearRegression()
LR = model.fit(X_Reshape, Y)

x_range = np.linspace(X_Reshape.min(), X_Reshape.max(), 20)
y_range = model.predict(x_range.reshape(-1, 1))
y_hat = lowess(Y, X)

fig = px.scatter(x = X, y = Y, opacity = 0.8, color_discrete_sequence=['black'])
fig.add_traces(go.Scatter(x = x_range, y=y_range, name='Linear Regression',
line=dict(color='limegreen')))
fig.add_traces(go.Scatter(x = y_hat[:,0], y=y_hat[:,1], name = 'LOWESS Smoothening',
line=dict(color='red')))
fig.update_layout(dict(plot_bgcolor = 'white'))
```

```
fig.update_xaxes(showgrid=True, gridwidth=1, gridcolor='lightgrey', zeroline=True,
zerolinewidth=1, zerolinecolor='lightgrey', showline=True, linewidth=1, linecolor='black')
fig.update_yaxes(showgrid=True, gridwidth=1, gridcolor='lightgrey', zeroline=True,
zerolinewidth=1, zerolinecolor='lightgrey', showline=True, linewidth=1, linecolor='black')
fig.update_layout(title=dict(text="LOWESS on DataSet", font=dict(color='black')))
fig.update_traces(marker=dict(size=3))
fig.show()
```

## OUTPUT

Performing necessary imports




Experiment for different point (.5 to .05) and tau (10 to 100)

```
def predict(X, y, point, tau):
    m = X.shape[0]
    X_ = np.append(X, np.ones(m).reshape(m,1), axis=1)
    point_ = np.array([point, 1])
    w = wm(point_, X_, tau)
    theta = np.linalg.pinv(X_.T*(w * X_))*(X_.T*(w * y))
    pred = np.dot(point_, theta)
    return theta, pred
```

✓ 0.3s

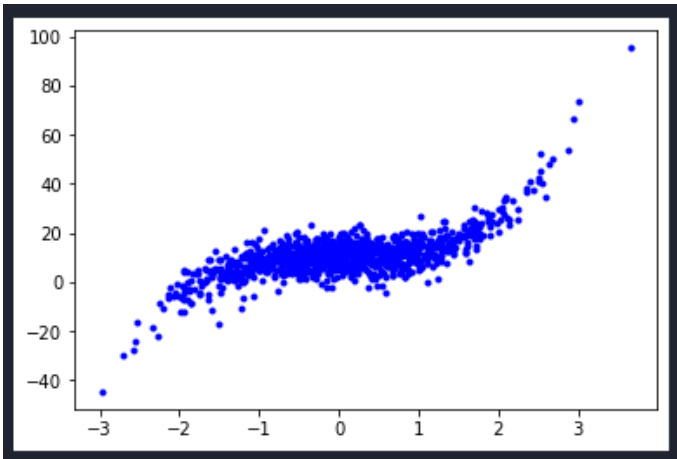
```
def plot_predictions(X, y, tau, nval):
    X_test = np.linspace(-3, 3, nval)
    preds = []
    for point in X_test:
        theta, pred = predict(X, y, point, tau)
        preds.append(pred)
    X_test = np.array(X_test).reshape(nval,1)
    preds = np.array(preds).reshape(nval,1)
    plt.plot(X, y, 'b.')
    plt.plot(X_test, preds, 'r.')
    plt.show()
```

✓ 0.3s

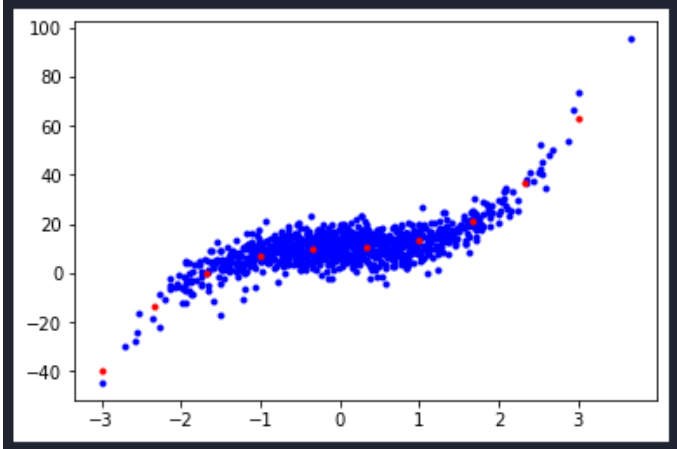
Plotting

```
X = np.random.randn(1000,1)
y = 2 * (X ** 3) + 10 + 4.6 * np.random.randn(1000, 1)
plt.plot(X, y, 'b.')
plt.show()

for i in [0.5, 0.1, 0.05]:
    for j in [10, 50, 100]:
        print("\n\nTau: {}\nPoints: {}".format(i, j))
        plot_predictions(X, y, i, j)
```

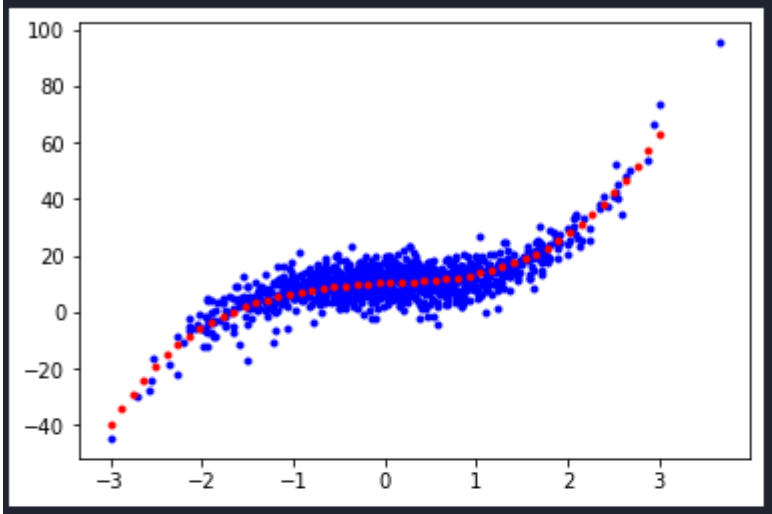
✓ 6.6s



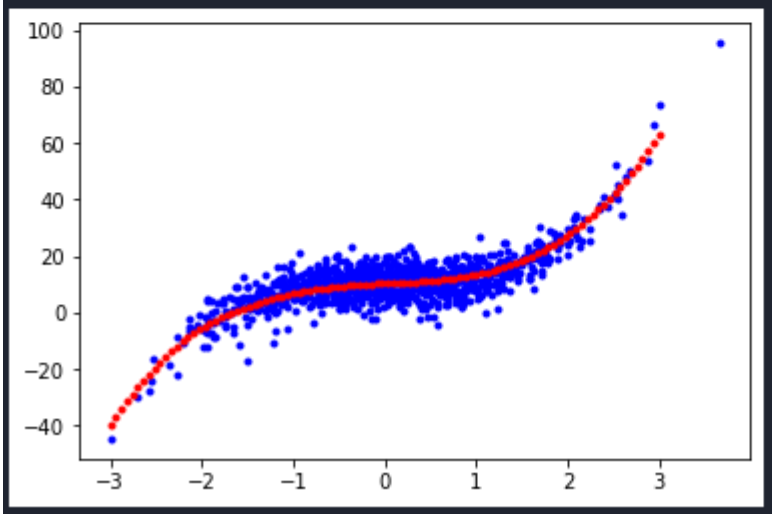

```
Tau: 0.5
Points: 10
```



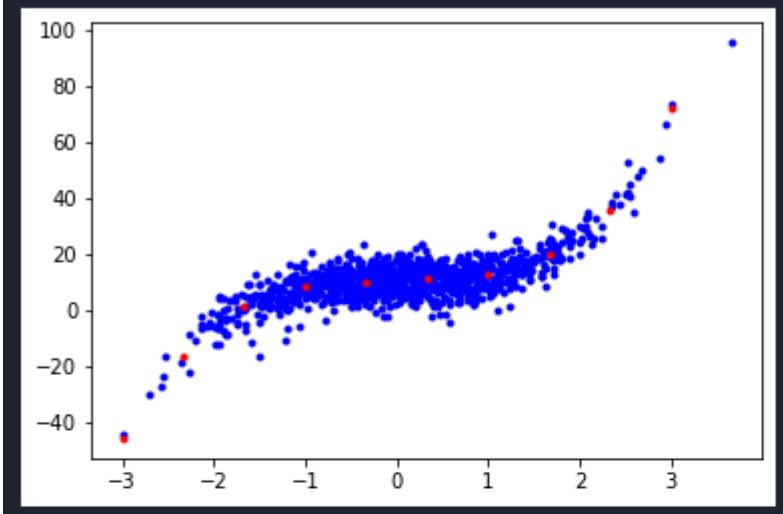
```
Tau: 0.5
Points: 50
```



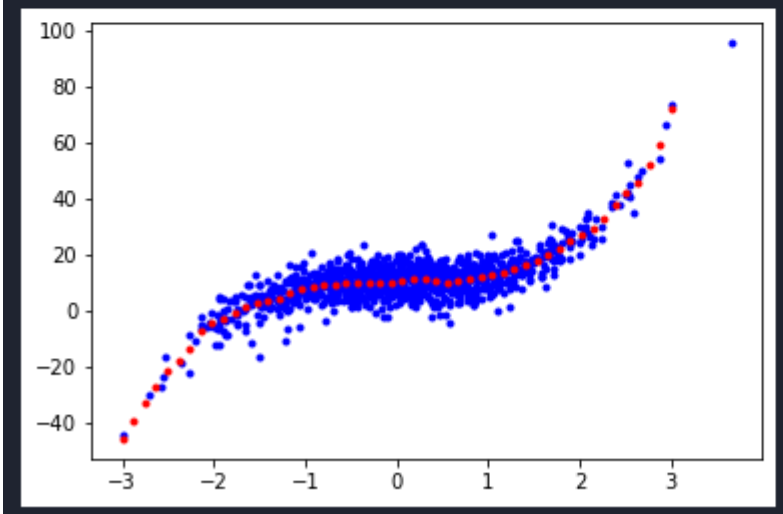

```
Tau: 0.5
Points: 100
```



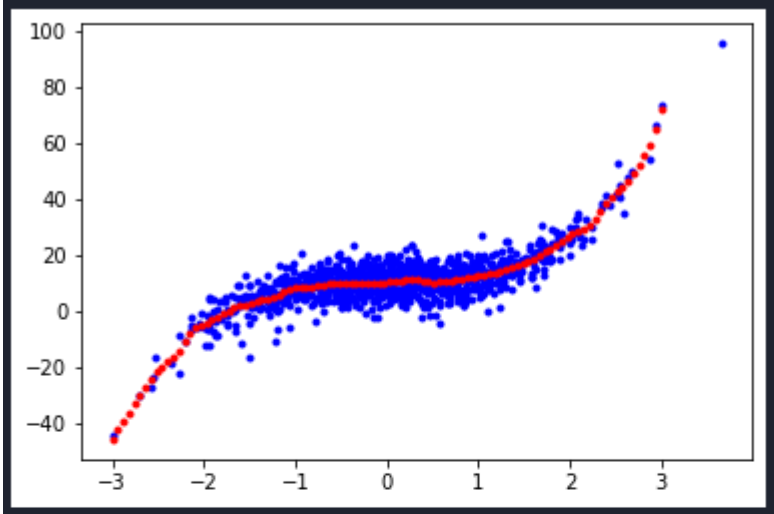
```
Tau: 0.1
Points: 10
```



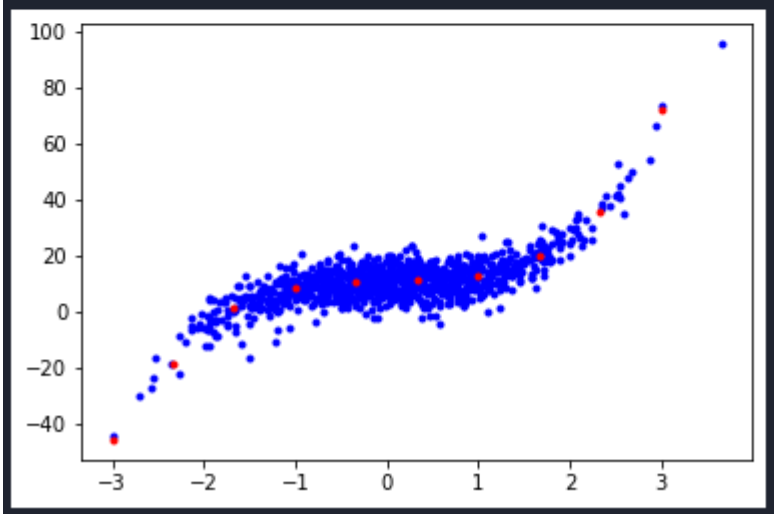

```
Tau: 0.1
Points: 50
```



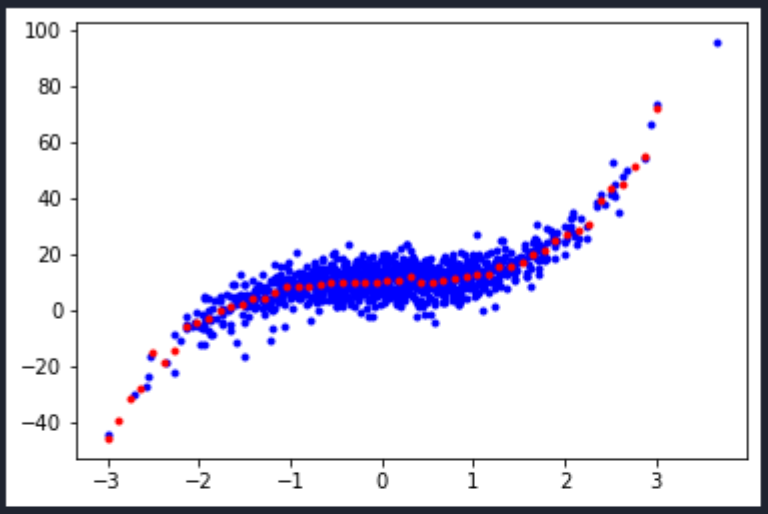
```
Tau: 0.1
Points: 100
```



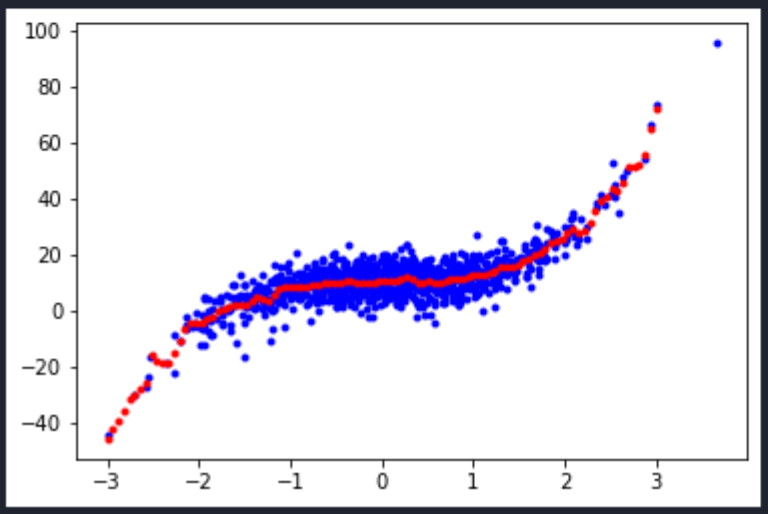

```
Tau: 0.05
Points: 10
```



```
Tau: 0.05
Points: 50
```




```
Tau: 0.05
Points: 100
```

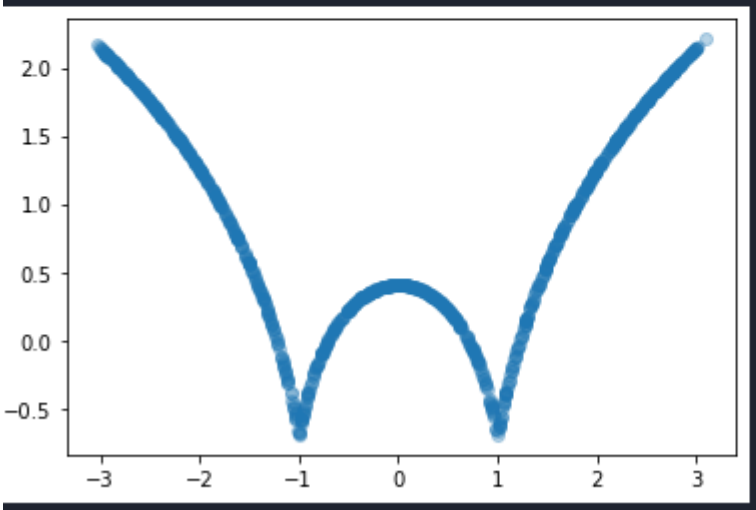


## Performing LOWESS

```
X = np.linspace(-3, 3, 1000)
print(X.shape)
X += np.random.normal(scale = 0.05, size = 1000)
Y = np.log(np.abs((X**2) - 1) + 0.5)
print(Y.shape)
plt.scatter(X, Y, alpha = 0.32)
plt.show()
```

✓ 0.1s

```
(1000,)
(1000,)
```




```
X_Reshape = X.reshape(-1, 1)
model = LinearRegression()
LR = model.fit(X_Reshape, Y)
```

✓ 0.3s

```
x_range = np.linspace(X_Reshape.min(), X_Reshape.max(), 20)
y_range = model.predict(x_range.reshape(-1, 1))
y_hat = lowess(Y, X)
```

✓ 0.3s

Plotting

```
fig = px.scatter(x = X, y = Y, opacity = 0.8, color_discrete_sequence=['black'])
fig.add_traces(go.Scatter(x = x_range, y=y_range, name='Linear Regression', line=dict(color='limegreen')))
fig.add_traces(go.Scatter(x = y_hat[:,0], y=y_hat[:,1], name = 'LOWESS Smoothening', line=dict(color='red')))
fig.update_layout(dict(plot_bgcolor = 'white'))
fig.update_xaxes(showgrid=True, gridwidth=1, gridcolor='lightgrey', zeroline=True, zerolinewidth=1, zerolinecolor='lightgrey', showline=True, linewidth=1, linecolor='black')
fig.update_yaxes(showgrid=True, gridwidth=1, gridcolor='lightgrey', zeroline=True, zerolinewidth=1, zerolinecolor='lightgrey', showline=True, linewidth=1, linecolor='black')
fig.update_layout(title=dict(text="LOWESS on DataSet", font=dict(color='black')))
fig.update_traces(marker=dict(size=3))
fig.show()
```

✓ 0.1s