NON-PARAMETRIC ESTIMATION

NAME – ABHIRUP BAG
ROLL – 13000122082
DEPARTMENT – CSE(B)
SEMESTER – 6
PAPER – PATTERN RECOGNITION(PEC-IT602D)

CONTENT

INTRODUCTION

- **What is Non-Parametric Estimation?**
  - ✓ *A statistical method that makes minimal assumptions about data distribution.*
  - ✓ *Unlike parametric methods, it does not assume a fixed functional form.*
- **Why is it Important?**
  - ✓ *Useful for analyzing real-world data with unknown or complex distributions.*
  - ✓ *Provides more flexibility in modeling.*

KEY CONCEPTS

- **No Fixed Parameters :** Unlike parametric methods (e.g., normal distribution with mean and variance), non-parametric methods do not assume a predefined shape for the data distribution.
- **Data-Driven Approach :** These methods rely on the structure of the observed data to make inferences.
- **More Flexible but More Data Needed :** Non-parametric methods adapt to data patterns but often require larger sample sizes to achieve similar accuracy as parametric methods.

COMMON NON-PARAMETRIC METHODS

Kernel Density Estimation (KDE)

Histogram Estimation

Nearest Neighbour Estimation

Empirical Cumulative Distribution Function (ECDF)

KERNEL DENSITY ESTIMATION (KDE)

▪ Definition :

A method to estimate a probability density function (PDF) by smoothing observed data.

Gaussian Kernel

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{u^2}{2}\right]$$

▪ Formula :

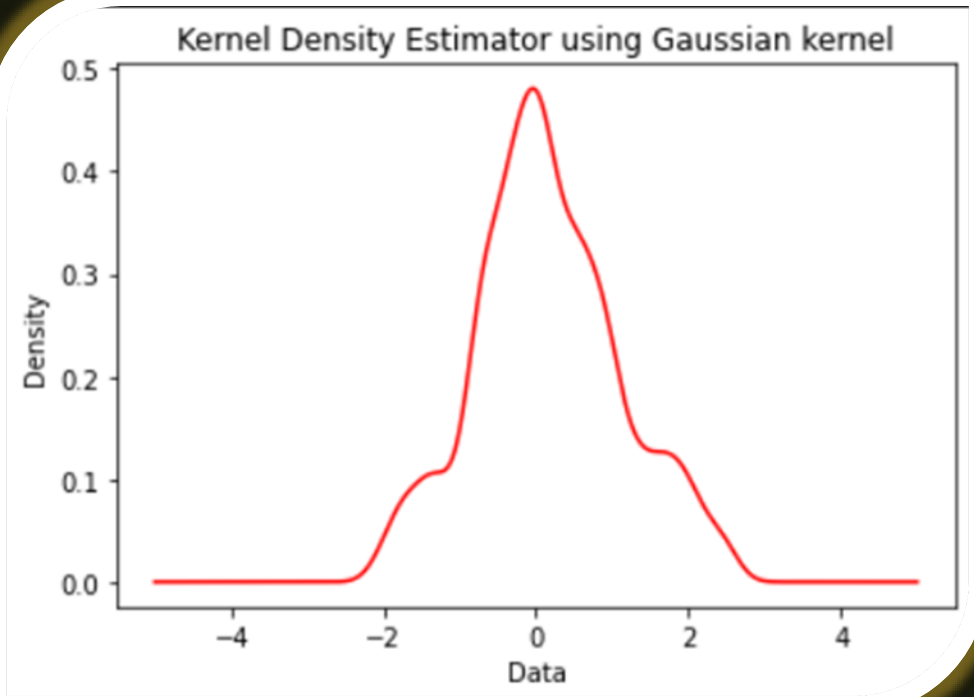
$$f(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{n}\right)$$

- K = Kernel function (e.g., Gaussian, Epanechnikov).
- h = Bandwidth (controls smoothness).

▪ Python Code :

```
import numpy as np
from scipy.stats import norm
from sklearn.neighbors import KernelDensity
# Kernel Density Estimator using gaussian kernel
X = np.random.randn(100)
model = KernelDensity(kernel='gaussian',
                      bandwidth=0.2)
model.fit(X[:, None])
new_data = np.linspace(-5, 5, 1000)
density = np.exp(model.score_samples(new_data[:, None]))
# print(new_data)
# Plot the densities
plt.plot(new_data, density, '-',
         color='red')
plt.xlabel('Data')
plt.ylabel('Density')
plt.title('Kernel Density Estimator using Gaussian kernel')
plt.show()
```

Output


Kernel Density Estimator using Gaussian kernel
Density
0.0
0.1
0.2
0.3
0.4
0.5
−4
−2
0
2
4
Data

HISTOGRAM ESTIMATION

What is a histogram?

- Splits the data range into bins and counts the frequency of values in each bin.
- A simple method to estimate probability density.

Pros and Cons :

- *Pros*: *Simple and easy to interpret.*
- *Cons*: *Highly sensitive to bin width selection.*

Histogram estimator

$$\hat{p}(x) = \frac{\#\{x^t \text{ in the same bin as } x\}}{Nh}$$

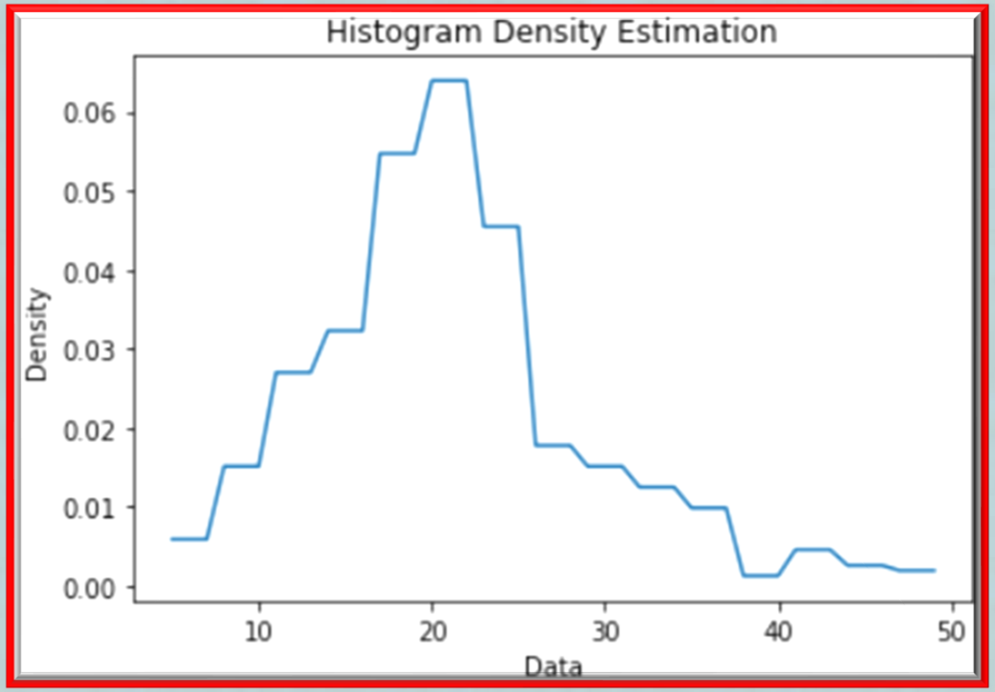
■ Python Code to plot a histogram

```
import numpy as np

def hist_pdf(x, data, n_bins=2,
             minv=None, maxv=None):
    if minv is None:
        minv = np.min(data)
    if maxv is None:
        maxv = np.max(data)
    d = (maxv-minv) / n_bins
    bins = np.arange(minv, maxv, d)
    bin_id = int((x-minv)/d)
    bin_minv = minv+d*bin_id
    bin_maxv = minv+d*(bin_id+1)
    n_data = len(data)
    y = len(data[np.where((data > bin_minv)\
                    & (data < bin_maxv))])
    pdf = (1.0/d) * (y / n_data)
    return pdf
```

```
from sklearn.datasets import load_boston
import matplotlib.pyplot as plt
ds = load_boston()
data = ds['target']
# Demo histogram
xvals = np.arange(min(data), max(data), 1)
n_bins = 15
pdf = [hist_pdf(x, data, n_bins=n_bins) for x in
xvals]
plt.xlabel('Data')
plt.ylabel('Density')
plt.title('Histogram Density Estimation')
plt.plot(xvals, pdf)
plt.show()
```

Output


Histogram Density Estimation
0.06
0.05
0.04
0.03
0.02
0.01
0.00
Density
10
20
30
40
50
Data

NEAREST NEIGHBOR ESTIMATION (K-NN DENSITY ESTIMATION)

What is Nearest Neighbor Estimation?

- Determines the density of a point based on the distance to its k-nearest neighbor.
- Provides a local density estimate based on proximity to other data points.

Strengths & Weaknesses:

✓ Works well for high-dimensional data.

✗ Computationally expensive, especially for large datasets.

Formula :

$$\hat{f}(x) = \frac{k}{nV}$$

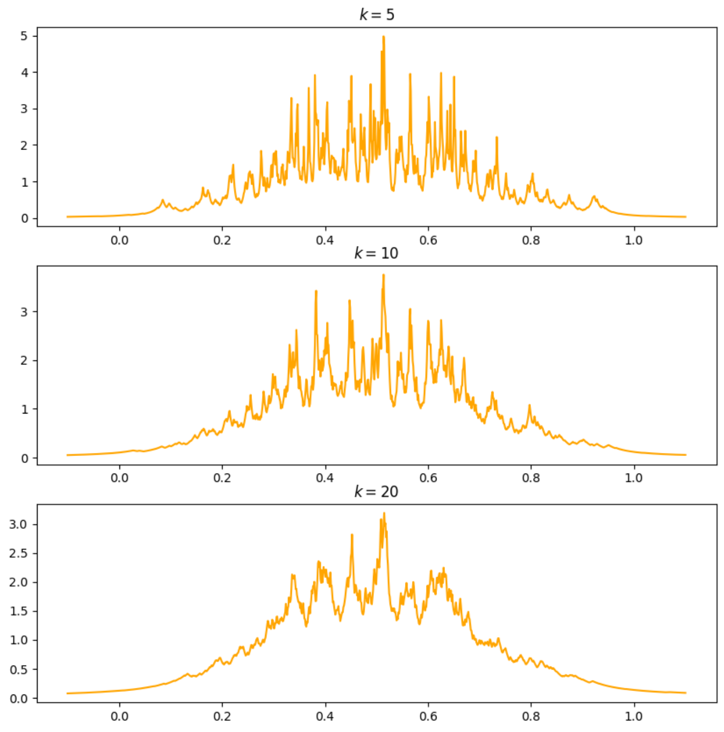
- V is the **volume** around xxx containing k nearest neighbors.

📌 **Example:** Used in anomaly **detection** (e.g., fraud detection in banking transactions).

■ Python Code to plot a K-NN DENSITY ESTIMATOR

```python
# KNN Density esitmator
gaussian = norm(loc=0.5, scale=0.2)
X = gaussian.rvs(500)
grid = np.linspace(-0.1, 1.1, 1000)
k_set = [5, 10, 20]
fig, axes = plt.subplots(3, 1, figsize=(10, 10))
for i, ax in enumerate(axes.flat):
    K = k_set[i]
    p = np.zeros_like(grid)
    n = X.shape[0]
    for i, x in enumerate(grid):
        dists = np.abs(X-x)
        neighbours = dists.argsort()
        neighbour_K = neighbours[K]
        p[i] = (K/n) * 1/(2 * dists[neighbour_K])
    ax.plot(grid, p, color='orange')
    ax.set_title(f'$k={K}$')
plt.show()
```

Output



EMPIRICAL CUMULATIVE DISTRIBUTION FUNCTION (ECDF)

What is ECDF?

- Estimates the cumulative probability of a variable.
- Shows how many observations are below or equal to a given value.

Formula :

$$\widehat{F}(x) = \frac{1}{n}\sum_{i=1}^{n} I(x_i \leq x)$$

- where $I(x_i \leq x)$ is an indicator function (1 if true, 0 otherwise).

Why is ECDF Useful?

✓ More informative than histograms.

✓ No need to choose bin width or bandwidth.

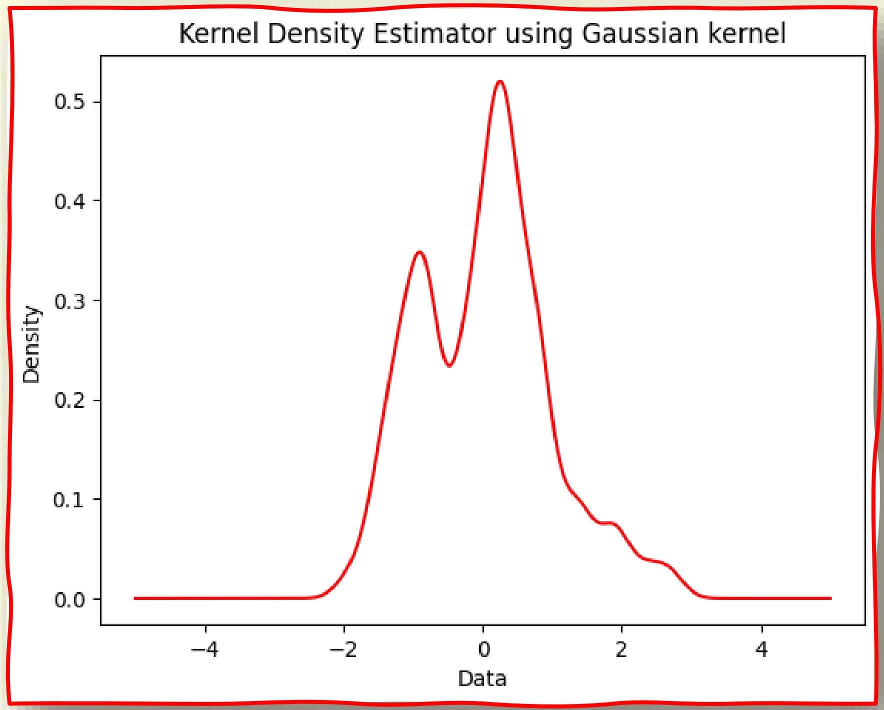
✓ Useful for comparing distributions (e.g., Kolmogorov-Smirnov test).

📌 Example: Income distribution comparison between two cities.

▪ Python Code to plot ECDF

```
import numpy as np
from scipy.stats import norm
from sklearn.neighbors import KernelDensity
# Kernel Density Estimator using gaussian kernel
X = np.random.randn(100)
model = KernelDensity(kernel='gaussian',
                      bandwidth=0.2)
model.fit(X[:, None])
new_data = np.linspace(-5, 5, 1000)
density = np.exp(model.score_samples(new_data[:, None]))
# print(new_data)
# Plot the densities
plt.plot(new_data, density, '-',
        color='red')
plt.xlabel('Data')
plt.ylabel('Density')
plt.title('Kernel Density Estimator using Gaussian kernel')
plt.show()
```

Output


Kernel Density Estimator using Gaussian kernel
0.5
0.4
0.3
0.2
0.1
0.0
Density
−4
−2
0
2
4
Data

SPLINE & LOCAL POLYNOMIAL REGRESSION (LOESS)

What is LOESS?

- A non-parametric regression technique that fits small polynomials locally.
- Useful for trend estimation in non-linear data.

Strengths:

✓ Captures complex patterns.

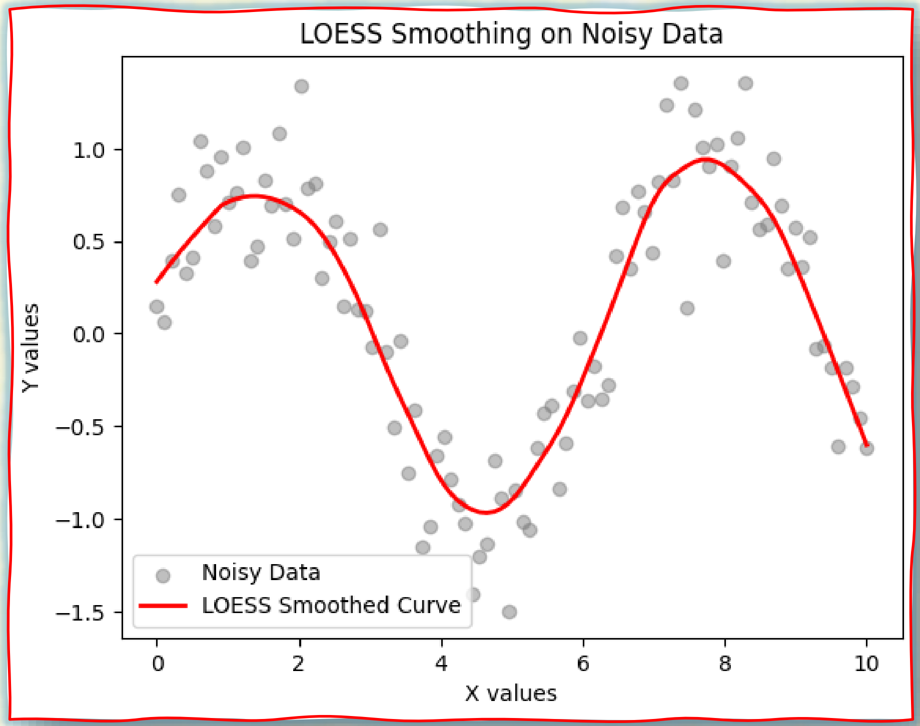
✓ Smoothens out local fluctuations.

📌 **Example: Used in stock price trends and climate modeling.**

Graph: A LOESS-smoothed curve applied to noisy data

```
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
# Generate Noisy Data
np.random.seed(42)
x = np.linspace(0, 10, 100)  # X values from 0 to 10
y = np.sin(x) + np.random.normal(0, 0.3, size=len(x))  # Add noise to sine function
# Apply LOESS Smoothing
lowess = sm.nonparametric.lowess(y, x, frac=0.2)  # frac controls smoothness
# Extract smoothed values
x_smooth, y_smooth = lowess[:, 0], lowess[:, 1]
# Plot Original Noisy Data
plt.scatter(x, y, label="Noisy Data", color="gray", alpha=0.5)
# Plot LOESS Smoothed Curve
plt.plot(x_smooth, y_smooth, label="LOESS Smoothed Curve", color="red", linewidth=2)
# Labels and Title
plt.xlabel("X values")
plt.ylabel("Y values")
plt.title("LOESS Smoothing on Noisy Data")
plt.legend()
plt.show()
```

Output



LOESS Smoothing on Noisy Data
Y values
X values
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
2
4
6
8
10
Noisy Data
LOESS Smoothed Curve

ADVANTAGES & DISADVANTAGES OF NON-PARAMETRIC ESTIMATION

✅ **Advantages:**

✓ No assumption about underlying distribution.
✓ More **adaptive and flexible** to real-world data.
✓ Works well for **high-dimensional and irregular data.**

❌ **Disadvantages:**

✗ **More data needed** for accurate estimation.
✗ **Computationally expensive**, especially for large datasets.
✗ Choice of **bandwidth/smoothing parameters** affects results.

📌 **Key Takeaway:** While flexible, non-parametric methods require careful tuning for accurate results.

APPLICATIONS

- 📌 **Density Estimation:** Income distribution, species population density.
- 📌 **Trend Analysis:** Stock prices, climate data.
- 📌 **Classification/Regression:** Medical diagnosis (k-NN), real estate pricing.
- 📌 **Hypothesis Testing:** Non-parametric tests (Mann-Whitney U test).
- 📌 **Visual:** Real-world examples (e.g., KDE plot of income data).

CONCLUSION

- ✓ Non-parametric estimation is powerful for analyzing unknown distributions.
- ✓ KDE, histograms, ECDF, and LOESS are key techniques.
- ✓ Choosing the right method depends on sample size, accuracy needs, and computation limits.

Thank you

ABHIRUP BAG

abhirup7477@gmail.com

ROLL NO. : 13000122082