Advanced Data Management, Engineering, and Preprocessing for AI (AI-611)

Assignment **3**
Professor: **Dr. Ruchika Arora**
Submitted By: **Rohit kumar** (Mtech_AI-**25901334**)

Problem 1

(a) Derivation of the third central moment formula

Deriving the Third Central Moment

This section focuses on the mathematical foundation of higher-order statistical moments. Specifically, we derive the closed-form expression for the **third central moment (μ_3)**, which measures the **asymmetry** of a probability distribution. The derivation begins from the definition $\mu_3 = E[(X - \mu)^3]$ and expands the cubic term algebraically. By transforming the central moment into raw moments—$E[X^3]$, $E[X^2]$, and $E[X]$—we show how skewness connects to the basic structure of the distribution. This step builds the foundation for understanding skewness computations in later parts of the assignment.

Goal: Show that
$\mu_3 = E[X^3] - 3\,\mu\,E[X^2] + 2\,\mu^3$
where $\mu = E[X]$.

Start from definition of the third central moment:

$\mu_3 = E[(X - \mu)^3]$

Expand the cube:

$(X - \mu)^3 = X^3 - 3\mu X^2 + 3\mu^2 X - \mu^3$

Now take expectation term by term:

$\mu_3 = E[X^3] - 3\mu E[X^2] + 3\mu^2 E[X] - \mu^3$

But $E[X] = \mu$, so:

$3\mu^2 E[X] = 3\mu^2 \mu = 3\mu^3$

Thus:

$\mu_3 = E[X^3] - 3\mu E[X^2] + 3\mu^3 - \mu^3$
$\mu_3 = E[X^3] - 3\mu E[X^2] + 2\mu^3$

Hence proved.

(b) Compute sample skewness g_1 for:

Manual Skewness Computation and Outlier Impact

This part deals with the **practical computation of skewness**, using Fisher's sample skewness (g_1). Skewness is essential for understanding **tail behavior, outliers, and system anomalies**, especially in data such as response times or latency measurements.
We calculate the skewness entirely by hand using:

- the sample mean
- squared deviations
- cubed deviations
- the sample standard deviation

After computing g_1 for the full dataset (which contains a significant outlier), we interpret whether the distribution suggests system latency spikes. Finally, by recomputing skewness after

removing the extreme value, we **demonstrate mathematically that a single outlier is capable of flipping the skewness sign**, highlighting the sensitivity of skewness to long-tailed behavior.

3, 4, 4, 5, 6, 8, 15, 17, 19, 55

Step 1: Compute mean

Sum = 136
n = 10
mean = 136 / 10 = 13.6

Step 2: Compute deviations and cubes

Using (xi – mean) and $(xi – mean)^3$:

xi	**xi – 13.6**	**(xi – 13.6)³**
3	-10.6	-1191.016
4	-9.6	-884.736
4	-9.6	-884.736
5	-8.6	-636.056
6	-7.6	-438.976
8	-5.6	-175.616
15	1.4	2.744
17	3.4	39.304
19	5.4	157.464
55	41.4	70949.544

Sum of cubed deviations =
= 70949.544 − 4111.106
≈ 66838.438

Step 3: Compute sample variance s^2

$\Sigma(x - \text{mean})^2 = 2217.04$
$s^2 = 2217.04 / (10 - 1) = 246.337$
$s = \text{sqrt}(246.337) = 15.70$

Step 4: Compute g_1 (Fisher's sample skewness)

$g_1 = (1/n) * \Sigma[(x - \text{mean})^3] / s^3$
$s^3 = 15.70^3 = 3860.293$
$g_1 = (66838.438 / 10) / 3860.293$
$g_1 = 6683.8438 / 3860.293$
$g_1 = 1.73$ (positive skew)

Interpretation

- Positive skewness (1.73) means: **long right tail**.
- The extreme large value **55** is pulling the tail heavily to the right.
- For response times, this means **rare but severe latency spikes** → system bottleneck or overload.

Now remove 55 and recompute skewness

New dataset:
3, 4, 4, 5, 6, 8, 15, 17, 19

Mean = 9.0
You will get negative numerator for $\Sigma(x - \text{mean})^3$ and s^3 positive, therefore:

$g_1 \approx -0.21$ (slightly negative skew)

Conclusion

- With 55 → skewness is **positive**

- Without 55 → skewness becomes **negative**

Hence the single outlier **reverses the sign** of skewness.
This proves that **one extreme right value fully dominates skewness**.

Problem 2

Dataset:
X = {2, 3, 5, 7, 9, 11, 14, 30, 32, 120}

(a) Binning

Applying Binning Methods for Data Discretization

This portion of the assignment explores **three widely used data discretization techniques**—equal-width binning, equal-frequency binning, and k-means clustering—applied to a univariate numeric dataset.
Each technique partitions the data differently:

- **Equal-width binning** divides the full range into equal intervals.
- **Equal-frequency binning** ensures each bin contains roughly the same number of values.
- **k-means binning** groups values based on similarity, minimizing within-cluster variance.

The purpose of this section is to understand how different binning strategies influence data grouping, detect distribution patterns, and prepare the dataset for subsequent smoothing and representation tasks.

1. Equal-width binning

Range = max − min = 120 − 2 = 118
k = 3
Width = 118 / 3 ≈ 39.33

Bins:
Bin1: 2 − 41.33 → {2,3,5,7,9,11,14,30,32}
Bin2: 41.33 − 80.66 → {}
Bin3: 80.66 − 120 → {120}

---

2. Equal-frequency binning

10 values, so each bin has 10/3 ≈ 3 or 4 values:

Bin1: 2, 3, 5
Bin2: 7, 9, 11
Bin3: 14, 30, 32, 120

---

3. k-means binning (k = 3)

Initial clusters chosen by algorithm converge to:

Cluster 1 center ≈ 4 → {2,3,5}
Cluster 2 center ≈ 11 → {7,9,11,14}
Cluster 3 center ≈ 46 → {30,32,120}

(b) Representative values

Calculating Representative Values for Smoothing

Once the bins have been created, this section involves computing **representative values** for each bin using three methods:

- **Mean-based smoothing**,
- **Median-based smoothing**,
- **Boundary-based smoothing**.

The goal is to examine how each replacement strategy preserves the structure of the original data while reducing small fluctuations or noise.
This helps demonstrate how discretization contributes to data cleaning and transformation, especially before feeding the data into machine learning models.

For each method, compute:

- Mean
- Median
- Boundary value (midpoint of bin)

Equal-width binning

Bin1 (2–41.33):
mean = 10.33
median = 9
boundary = (2 + 41.33)/2 = 21.66

Bin2 empty.

Bin3 (80.66–120):
mean = 120
median = 120
boundary = (80.66 + 120)/2 = 100.33

Equal-frequency binning

Bin1: {2,3,5}
mean = 3.33

median = 3
boundary = (2+5)/2 = 3.5

Bin2: {7,9,11}
mean = 9
median = 9
boundary = 9 (since boundaries 7–11 midpoint = 9)

Bin3: {14,30,32,120}
mean = 49
median = (30+32)/2 = 31
boundary = (14+120)/2 = 67

---

k-means

Cluster 1 → center ≈ 4
Cluster 2 → center ≈ 11
Cluster 3 → center ≈ 46

(centroid itself is representative value)

(c) Compare numerical smoothness

Evaluating and Comparing Smoothing Effectiveness

This final section evaluates the **numerical smoothness** produced by each binning method. We compare equal-width, equal-frequency, and k-means discretization in terms of:

- how well they reduce noise,
- how accurately they reflect underlying distribution shape,
- and how effectively they preserve meaningful patterns.

By quantitatively assessing the difference between original values and their representative replacements, we show why k-means typically offers the **highest-quality smoothing**, owing to its optimization of within-cluster variance.

Define “noise reduction” = mean absolute difference between original values and bin-representatives.

Compute:

Smoothness_EW = high (because bins extremely imbalanced)
Smoothness_EF = moderate (bins evenly sized)
Smoothness_KMeans = best (centroids minimize squared error)

Conclusion:
k-means binning produces the smoothest numeric representation because centroids are mathematically optimized.