Chapter - III

Discretization

Syllabus: Variable Discretization: - divide the variables into equal intervals, perform discretization followed by categorical encoding. Working with outliers: - Interpretation of outliers, trimming outliers, capping the variables at arbitrary max and min values, performing zero coding.

- ✓ In **Exploratory Data Analysis (EDA)**, **discretization** is the process of converting continuous numerical variables into discrete bins or categories.
- ✓ It's like grouping continuous values into labelled "buckets" to simplify patterns and make relationships easier to detect.

Why Discretization is Used in EDA

- ✓ **Simplifies interpretation** – Easier to explain patterns in categories than in raw continuous numbers.
- ✓ **Handles non-linearity** – Some models or plots work better with categorical values.
- ✓ **Reduces noise** – Small variations in continuous data are smoothed out.
- ✓ **Helps visualization** – Histograms, bar plots, and grouped analysis become more meaningful.
- ✓ **Improves model performance** – Certain algorithms (e.g., Naïve Bayes) perform better with categorical inputs.

Types of Discretization

1. Unsupervised Discretization
    a. Equal-width binning
    b. Equal-frequency binning

2. Supervised Discretization
    a. Decision tree-based binning
    b. ChiMerge / Chi2-based
    c. Entropy-based binning

1. Equal-width binning

- ✓ Equal-width binning is a **quantization** technique where the range of a continuous feature is divided into **intervals of the same width**.
- ✓ Every value in that range falls into one and only one bin.

- ✓ Think of it as slicing a ruler into equal-sized segments, regardless of how many points fall into each segment.

Mathematical Breakdown

Equal-width binning splits the numeric range into **k** equal intervals.

Step 1 – Bin Width Calculation

$$W = \frac{\text{Max} - \text{Min}}{k}$$

Where:

- W = bin width
- Max = maximum value in data
- Min = minimum value in data
- k = number of bins

Step 2 – Bin Boundaries

The bin boundaries are:

$$B_i = \text{Min} + i \times W, \quad \text{for } i = 0, 1, \ldots, k$$

So, you get:

$$B_0 = \text{Min}$$
$$B_1 = \text{Min} + W$$
$$B_2 = \text{Min} + 2W$$

Example:

Data = [6, 8, 15, 17, 20, 25, 50, 100]

Number of bins (k) = 3

Step 1 – Find Min, Max, Width

$$\text{Min} = 6, \quad \text{Max} = 100$$
$$\text{Width} = \frac{\text{Max} - \text{Min}}{k} = \frac{100 - 6}{3} = \frac{94}{3} \approx 31.33$$

Step 2 – Bin Boundaries

$$B_i = \text{Min} + i \times \text{Width}$$

We get:

- Bin 1: $6 \rightarrow 6 + 31.33 = 37.33$
- Bin 2: $37.33 \rightarrow 37.33 + 31.33 = 68.66$
- Bin 3: $68.66 \rightarrow 100$

Step 3 – Assign Each Value

Bin index formula:

$$\text{Index} = \left\lfloor \frac{x - \text{Min}}{\text{Width}} \right\rfloor$$

- 6: $\lfloor (6-6)/31.33 \rfloor = 0 \rightarrow$ Bin 1
- 8: $\lfloor (8-6)/31.33 \rfloor = 0 \rightarrow$ Bin 1
- 15: $\lfloor (15-6)/31.33 \rfloor = 0 \rightarrow$ Bin 1
- 17: Bin 1
- 20: Bin 1
- 25: Bin 1
- 50: $\lfloor (50-6)/31.33 \rfloor = 1 \rightarrow$ Bin 2
- 100: $\lfloor (100-6)/31.33 \rfloor = 3 \rightarrow$ capped to Bin 2 (final bin)

Result:

- Bin 1 (6–37.33): [6, 8, 15, 17, 20, 25] (6 values)
- Bin 2 (37.33–68.66): [50] (1 value)
- Bin 3 (68.66–100): [100] (1 value)

Advantageous, disadvantageous and applications of Equal-width binning

I. Advantages

1. **Simple & Easy to Implement**
   - Just need the min and max values to calculate bin width.
2. **Computationally Fast**
   - Works well even on large datasets because it's just basic arithmetic.
3. **Uniform Range Representation**
   - Each bin covers the same interval, making comparisons straightforward.
4. **Good for Normally Distributed Data**
   - If data is evenly spread, bins will have reasonable counts.

II. Disadvantages

5. **Uneven Data Distribution**
    - If data is skewed, some bins may have too few or too many values.
6. **Empty Bins Possible**
    - Gaps in data can lead to bins with zero observations.
7. **Poor Handling of Outliers**
    - Outliers can stretch bin ranges, making the main data lumped into fewer bins.
8. **Loss of Data Granularity**
    - Original precision is lost; values are grouped into broad categories.
9. **Not Adaptive to Density**
    - Doesn't account for where data points are concentrated.

III. Applications

1. **Exploratory Data Analysis (EDA)**
    - To quickly understand data distribution patterns.
2. **Feature Engineering**
    - Convert continuous variables into categorical bins for certain ML models.
3. **Customer Segmentation**
    - Group customers by age, income, or purchase frequency.
4. **Risk Categorization**
    - Classify loan applicants into low, medium, and high-risk bands.
5. **Sales/Marketing Analysis**
    - Break product prices into equal ranges for pricing strategies.
6. **Data Visualization**
    - Histograms with equal-width bins give an easy-to-read overview.

2. Equal frequency discretization

Equal Frequency Discretization (also called Equal-Frequency Binning, Quantile Binning, or Rank-based Binning) is a data transformation technique where:

- We convert continuous numerical data into categorical bins.
- Each bin has (almost) the same number of observations.
- Bin edges are determined based on data distribution rather than fixed width.

It's widely used in:

- Exploratory Data Analysis (EDA)

- Feature Engineering
- Credit Scoring
- Rank-based categorization

How it Differs from Equal-Width Binning

Aspect	Equal-Width Binning	Equal-Frequency Binning
Bin Size	Fixed value range for each bin	Fixed number of data points per bin
Distribution Sensitivity	Ignores data distribution	Sensitive to data distribution
Outlier Handling	Can create many empty bins for skewed data	Automatically adjusts bin edges
Example	Age: 0–20, 20–40, 40–60…	First 25%, next 25%, etc.

The Core Algorithm

Given:

- $X = \{x_1, x_2, \ldots, x_N\}$ → continuous numerical variable
- k = number of bins

Step-by-step:

1. **Sort** the data: X_{sorted}
2. **Compute bin size**:

$$n_{bin} = \frac{N}{k}$$

   (If N not divisible by k, distribute remainder across bins)

3. **Split into bins**:
   - Bin 1: first n_{bin} values
   - Bin 2: next n_{bin} values

   ...
4. **Assign bin labels**:
   - Either numeric (0, 1, 2, ...)
   - Or descriptive ("Low", "Medium", "High")

4. Mathematical Perspective (Quantiles)

EFD can be implemented via **quantile cut points**.

If we want **k bins**:

- Quantile boundaries =

$$Q_i = \text{Quantile}\left(\frac{i}{k}\right) \quad \text{for } i = 0, 1, ..., k$$

- Example for 4 bins: boundaries = $Q_0, Q_{0.25}, Q_{0.5}, Q_{0.75}, Q_1$ (quartiles).

5. Example Calculation

Dataset:
`[3, 7, 4, 12, 9, 15, 6, 8, 10, 20]`, $k = 3$

Step 1: Sort → `[3, 4, 6, 7, 8, 9, 10, 12, 15, 20]`
Step 2: Bin size = $10/3 \approx 3.33$
Step 3: Assign bins:

- Bin 1: `[3, 4, 6]`
- Bin 2: `[7, 8, 9]`
- Bin 3: `[10, 12, 15, 20]`

Here, each bin has ~same number of elements (3–4).

Advantages EFD

1. ***Balanced representation*** — ensures all bins are well-populated.
2. ***Good for skewed data*** — bins adapt to data spread.
3. ***Percentile-based*** — useful for ranking or percentile analysis.
4. ***Handles outliers gracefully*** — they go into the top or bottom bin without distorting others.

Disadvantages EFD

1. ***Unequal bin widths*** — value ranges may differ drastically.
2. ***Splitting identical values*** — possible if many duplicates fall at a boundary.
3. ***Less interpretable for continuous ranges*** — "Bin 2" might mean 25–90 in one dataset but 60–65 in another.
4. ***Boundary instability*** — adding/removing a few data points can change bin edges significantly.

Applications of EFD

1. ***Customer segmentation:*** e.g., divide into top/middle/bottom spenders.
2. ***Credit scoring:*** income, loan amount, repayment times.
3. ***Sports analytics:*** rank players into tiers by performance stats.
4. ***Healthcare:*** group patients by percentile of age or lab results.
5. ***Education:*** grade students into percentile-based performance groups.

Decision Tree Binning

Decision Tree Binning is a **supervised discretization method** where a **decision tree algorithm** is used to split a continuous variable into **bins (intervals)** based on its relationship with the **target variable**.

- **Normal binning (equal-width/equal-frequency)**: Just cuts numbers into fixed ranges, without caring about the target.
- **Decision tree binning**: Finds **optimal split points** (thresholds) that best separate the target classes.

How it works?

1. **Take a continuous feature**
   Example: Age, Income, Salary, Marks.

2. **Choose the target variable (y)**
   - The target can be binary (Yes/No, Default/Not Default, Disease/No Disease) or categorical.
   - This is what the decision tree will try to predict.

3. **Fit a Decision Tree on (X=feature, y=target)**
   - Train a Decision Tree Classifier (for classification) or Decision Tree Regressor (for regression).
   - The tree tries to split the continuous feature into intervals (thresholds) that maximize the difference in target classes.
   - It uses metrics like Gini Index, Entropy, or Information Gain to decide the best split point.

4. **Find the split points (thresholds)**
   - The decision tree will automatically find the best places to cut the continuous variable.
   - Each split divides the data into two groups that are as different as possible with respect to the target.

   Example:
   For Age vs Default
   Split at Age = 30
   Split at Age = 50
   So bins become:
   ≤ 30, 31–50, > 50

5. **Assign each value to a bin**
   - Each interval becomes a bin/category.
   - Instead of using raw values, you use the bin ID (like Bin1, Bin2, Bin3).

Example: Student Marks vs Pass/Fail

Student	Marks	Pass (1=Yes, 0=No)
S1	10	0
S2	20	0
S3	25	0
S4	35	1
S5	50	1
S6	70	1

Step 1: Raw Feature
The feature is Marks (continuous).

Step 2: Train a Decision Tree
The tree checks where Pass/Fail changes most clearly.
It sees that students with Marks ≤ 30 usually fail, and those with Marks > 30 usually pass.

Step 3: Create Bins
- Bin1: Marks ≤ 30 → Fail Zone
- Bin2: Marks > 30 → Pass Zone

Step 4: Transform Data

Student	Marks	Pass	Marks_Bin
S1	10	0	Bin1
S2	20	0	Bin1
S3	25	0	Bin1
S4	35	1	Bin2
S5	50	1	Bin2
S6	70	1	Bin2

Use Cases of Decision Tree Binning

1. Credit Scoring & Risk Modeling

- Continuous variables like Age, Income, Loan Amount, Credit Balance are binned using decision trees.
- Example: Banks use it to decide if someone is a low, medium, or high-risk borrower.
- Helps in regulatory reporting (because categories are easier to explain than raw numbers).

2. Healthcare & Medical Analytics

- Features like BMI, Blood Pressure, Cholesterol, Age are binned into risk groups.
- Example:

  - Age ≤ 20 → Low risk
  - 20 < Age ≤ 40 → Medium risk
  - Age > 40 → High risk of disease
- Doctors and researchers use these bins for risk stratification.

3. Fraud Detection
- Continuous variables like transaction amount, frequency, or account age are binned.
- Example:
  - Transaction ≤ ₹500 → Safe
  - ₹500–₹5000 → Normal
  - ₹5000 → Suspicious
- Makes it easier to detect fraud patterns.

4. Customer Segmentation & Marketing
- Retailers use income, spending score, purchase frequency.
- Example:
  - Income ≤ 30k → Budget customers
  - 30k–70k → Regular customers
  - 70k → Premium customers
- This helps in personalized marketing and product targeting.

5. Education Analytics
- Marks, attendance percentage, study hours can be binned.
- Example:
  - Marks ≤ 30 → Fail zone
  - 30–60 → Average
  - 60 → Excellent
- Makes it easier for educators to track student performance categories.

Advantages of Decision Tree Binning

1. **Target-aware**
   - Unlike equal-width or equal-frequency binning, decision tree binning considers the target variable when creating bins.
   - This means bins actually help in prediction.

2. **Captures Non-linear Relationships**
   - Many variables (like Age vs Disease risk) do not have a simple linear relationship.
   - Decision trees can cut in non-linear ways to capture patterns.

3. **Better Model Performance**
   - Helps algorithms like Logistic Regression, Naive Bayes, and Linear Models by providing categorical bins that fit the target distribution.

4. **Interpretability**
   - Bins (like Age ≤ 30, 30–50, >50) are easy to explain to business users or regulators compared to raw numbers.

5. **Reduces Noise**
   - Continuous variables may have noisy fluctuations; binning smooths them into meaningful groups.

Disadvantages of Decision Tree Binning

1. **Overfitting Risk**
   - If not controlled (too many splits), the bins may become very small and model-specific (memorizing the training set).
2. **Data Hungry**
   - Needs enough data points for the tree to find **reliable splits**. Small datasets may lead to unstable bins.
3. **Computationally More Expensive**
   - Compared to simple equal-width/frequency binning, decision tree binning requires **training a model** (extra time).
4. **Different Results Each Time**
   - If data changes slightly, the splits may shift (especially without constraints like max depth).
5. **Harder for Unsupervised Tasks**
   - Since it requires a **target variable**, it cannot be used in pure clustering or unsupervised problems.

Chi-square (χ^2) based binning

It is a supervised binning (discretization) method that uses the Chi-square statistical test to merge continuous values into meaningful intervals (bins), based on how well they relate to the target variable.

How it works:

1. **Start with each distinct value of the feature as its own bin.**
   Example: Marks = [10, 20, 25, 35, 50, 70] → bins = [10], [20], [25], [35], [50], [70]

2. **For each pair of adjacent bins**, perform a **Chi-square test** on the target variable (Pass/Fail).
   - Null hypothesis: "These two bins are similar (no significant difference)."
   - If χ^2 value is small → merge them.
   - If χ^2 value is large → keep them separate.

3. **Repeat merging until:**
   - A stopping threshold is reached (like p-value or number of bins).

Marks	Pass
10	0
20	0
25	0
35	1
50	1
70	1

- Initially, each unique mark is its own bin.
- Compare **[10,20] vs [25]** → both are mostly Fail → merge them.
- Compare **[25] vs [35]** → Fail vs Pass → keep separate.
- Finally, we may end up with bins like:
  - ✓ Bin1: [10,20,25] → Fail
  - ✓ Bin2: [35,50,70] → Pass

Advantages

1. Captures statistical relationship between feature and target.
2. Produces stable bins guided by significance testing.
3. Good for credit scoring.

Disadvantages

1. Computationally expensive for large data.
2. Needs categorical target (binary or multi-class).
3. Might merge too much or too little depending on threshold.

Applications

1. Credit Risk modeling (common in banking).
2. Feature engineering for logistic regression.
3. Reducing cardinality of continuous variables.
4. Often combined with WoE (Weight of Evidence) transformation in finance.

Entropy Based Binning:

Entropy-based binning is a supervised discretization technique used in Exploratory Data Analysis (EDA) to convert continuous numerical features into discrete bins or intervals using class label information.

What is Entropy-Based Binning?

- ✓ It uses **entropy** to measure the **impurity** or **disorder** in a dataset.
- ✓ The goal is to **minimize entropy** within each bin — i.e., each bin should contain mostly one class label.
- ✓ Similar to how decision trees (like ID3 or C4.5) decide splits using information gain.

- **Entropy** of a dataset S:

$$H(S) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

where p_i is the proportion of class i in dataset S

- **Information Gain** from a split:

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

Here, S_v are the subsets formed by splitting on attribute A, and the goal is to maximize IG.

Why Use Entropy-Based Binning in EDA?

- ✓ Helps you **understand relationships** between numerical features and the target class.

- ✓ Enables **better visualizations** (e.g., stacked bar charts by bins).
- ✓ Useful for **feature engineering** or when using models that require categorical input (e.g., Naive Bayes).
- ✓ Often improves **predictive performance** by capturing non-linear relationships.

Example: Entropy-Based Binning

Age	Outcome
22	No
25	No
28	Yes
30	No
34	Yes
38	Yes
45	Yes
50	Yes

You want to discretize "Age" based on how it predicts "Outcome". Entropy-based binning will find the best cut (e.g., between 28 and 30) to group values that are predominantly "Yes" or "No".

Pros and Cons

Pros:

1. Uses label info (supervised)
2. Produces bins that help with classification
3. Often more meaningful than arbitrary binning

Cons:

1. More complex and computationally expensive
2. Only works when class labels are available
3. Can overfit if not regularized

```
!pip install mdlp-discretization

from sklearn.datasets import load_iris
from mdlp.discretization import MDLP
from sklearn.preprocessing import LabelEncoder

# Example with Iris
data = load_iris()
X = data.data
y = data.target

discretizer = MDLP()
X_disc = discretizer.fit_transform(X, y)
```

Working with outliers

An outlier is a data point that is significantly different or distant from the other observations in a dataset. It can be unusually high or low compared to the majority of the data.

Example:

Suppose you have the following ages of people in a group:

[22, 23, 24, 25, 23, 24, 22, 120]

- Most ages are between 22 and 25.
- But **120** is very different and much larger than the rest.

120 is an **outlier** because it's far away from the typical ages in the group.

Significance of working with outliers

1. **Improves Data Quality**
   Handling outliers helps identify and correct errors or inconsistencies in your data, ensuring more reliable analysis.
2. **Enhances Model Performance**
   Removing or adjusting outliers can prevent models from being biased or distorted, leading to better predictions and generalization.
3. **Prevents Misleading Conclusions**
   Outliers can skew summary statistics like mean and standard deviation. Managing them ensures your insights accurately represent the data.
4. **Reveals Important Insights**
   Instead of blindly removing outliers, analyzing them can uncover rare but critical events (e.g., fraud detection, equipment failure).
5. **Supports Robust Statistical Tests**
   Many statistical methods assume normally distributed data; outliers violate this assumption, so addressing them maintains test validity.
6. **Facilitates Better Visualization**
   Outliers can distort plots and charts. Handling them improves visual clarity and interpretability.

Interpretation of Outliers

When outliers are in data, it's important to **understand what they mean** rather than just removing them blindly. Here's how to interpret outliers:

1. **Are they Data Errors?**
   - Could the outlier be caused by mistakes in data entry, measurement errors, or faulty sensors?
   - If yes, correcting or removing them improves data quality.
2. **Are they Natural Variations?**
   - Some outliers may be valid but rare values naturally occurring in the data (e.g., very high income, extreme weather events).

- These might carry important information and shouldn't be discarded without thought.

3. **Do they Indicate Something Interesting?**
   - Outliers might point to fraud, system failures, or exceptional behavior worth investigating.
   - They can lead to new insights or domain discoveries.
4. **Impact on Analysis and Models**
   - Determine how much these outliers influence your summary statistics or model predictions.
   - Sometimes outliers disproportionately affect the mean or variance.
5. **Context Matters**
   - Always interpret outliers within the **domain context**. For example, a $1,000,000 sale may be an outlier in a small store but normal in luxury car sales.

Trimming outliers

Trimming outliers means removing the outlier data points entirely from your dataset. Instead of modifying or keeping them, you simply exclude these extreme values from your analysis.

Why Trim Outliers?

1. To reduce the impact of extreme values that can distort statistical measures like mean and standard deviation.
2. To improve model performance by training on cleaner, more representative data.
3. Useful when outliers are due to errors or noise rather than meaningful data.

Example:

Dataset:

$$7, 15, 36, 39, 40, 41, 42, 43, 47, 49$$

Step 1: Order the Data

The data is already ordered:

$$7, 15, 36, 39, 40, 41, 42, 43, 47, 49$$

Step 2: Find the Median (Q2)

The median splits the data into two halves.

- Number of data points $n = 10$
- Median position $M = \frac{n+1}{2} = \frac{10+1}{2} = 5.5$ (between 5th and 6th points)

Median Q_2 is the average of 5th and 6th values:

$$Q_2 = \frac{x_5 + x_6}{2} = \frac{40 + 41}{2} = 40.5$$

Step 3: Find the First Quartile (Q1)

Q_1 is the median of the **lower half** (all values below Q_2):

Lower half:

$$7, 15, 36, 39, 40$$

- $n_{lower} = 5$ (odd number of points)
- Median position $M_{lower} = \frac{n_{lower}+1}{2} = \frac{5+1}{2} = 3$

So,

$$Q_1 = x_3 = 36$$

Step 4: Find the Third Quartile (Q3)

Q_3 is the median of the **upper half** (all values above Q_2):

Upper half:

$$41, 42, 43, 47, 49$$

- $n_{upper} = 5$
- Median position $M_{upper} = 3$

So,

$$Q_3 = x_3 = 43$$

Step 5: Calculate the Interquartile Range (IQR)

$$IQR = Q_3 - Q_1 = 43 - 36 = 7$$

Step 6: Calculate Outlier Boundaries

- **Lower bound**:

$$\text{Lower Bound} = Q_1 - 1.5 \times IQR = 36 - 1.5 \times 7 = 36 - 10.5 = 25.5$$

- **Upper bound**:

$$\text{Upper Bound} = Q_3 + 1.5 \times IQR = 43 + 1.5 \times 7 = 43 + 10.5 = 53.5$$

Step 7: Identify Outliers

- Any data point < 25.5 or > 53.5 is considered an **outlier**.

Check dataset:

$$7, 15, 36, 39, 40, 41, 42, 43, 47, 49$$

- Below 25.5: **7, 15** → Outliers
- Above 53.5: None

```
import pandas as pd

data = pd.Series([10, 12, 15, 14, 13, 1000, 11, 13, 15, 9])

Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

trimmed_data = data[(data >= lower_bound) & (data <= upper_bound)]
print(trimmed_data)
```

Capping the variables at arbitrary max and min values

What is Capping?

Capping is a technique to **handle outliers** or extreme values in a dataset by **limiting (clipping) the values of a variable** within a specified range:

- ✓ Any value **greater than a chosen maximum threshold** is replaced by that maximum threshold.
- ✓ Any value **less than a chosen minimum threshold** is replaced by that minimum threshold.
- ✓ Values within the thresholds remain unchanged.

This process is often referred to as **Winsorizing** when applied to data.

Why Do We Use Capping?

Outliers can:

- ✓ Skew statistical measures like mean and standard deviation.
- ✓ Unduly influence machine learning models, especially those sensitive to extreme values.
- ✓ Affect the assumptions behind statistical tests.

Capping reduces the effect of these extreme values without deleting the data points entirely, preserving the size of the dataset and maintaining information.

When to Use Capping?

- ✓ When you want to **mitigate the impact of extreme outliers** but cannot or do not want to remove those records.
- ✓ When you have **domain knowledge** to define reasonable bounds for variables.
- ✓ When you want to **limit skewness** in your data.

- ✓ When the data contains errors or anomalies but you prefer adjusting rather than removing data points.

How to Choose Cap Values?

1. Domain Knowledge:

For example:

- Age should realistically be between 0 and 120.
- Temperature sensors may only operate between -50°C and 150°C.

Setting caps outside these boundaries prevents physically impossible values.

2. Statistical Percentiles:

- Set the minimum cap to the **1st percentile** (value below which 1% of data falls).
- Set the maximum cap to the **99th percentile**.

This approach assumes that the most extreme 1% on both tails are considered outliers.

3. Business or Operational Limits:

E.g., Credit card transaction limits, sensor output ranges, or financial thresholds.

Mathematical Definition

For a variable X, with chosen minimum cap $C_{\min}$ and maximum cap $C_{\max}$:

$$X_{\text{capped}} = \begin{cases} C_{\max} & \text{if } X > C_{\max} \\ C_{\min} & \text{if } X < C_{\min} \\ X & \text{otherwise} \end{cases}$$

Step-by-Step Example

Dataset:

$$[10, 12, 15, 14, 13, 1000, 11, 13, 15, 9]$$

Suppose we choose:

- $C_{\min} = 10$
- $C_{\max} = 20$

Apply capping:

Original	Capped
10	10
12	12
15	15
14	14
13	13
1000	20
11	11
13	13
15	15
9	10

Pros	Cons
Controls influence of extreme values without deletion	Choice of caps may be arbitrary or subjective
Preserves dataset size	Can distort data distribution
Simple to implement	May mask true variability or signal
Reduces skewness and impact on statistical measures	Over-capping may remove meaningful information

```
import pandas as pd

data = pd.Series([10, 12, 15, 14, 13, 1000, 11, 13, 15, 9])

# Define caps
min_cap = 10
max_cap = 20

# Apply capping
capped_data = data.clip(lower=min_cap, upper=max_cap)

print("Original data:\n", data)
print("\nCapped data:\n", capped_data)
```

When Should You Avoid Capping?

- ✓ When outliers represent important or meaningful phenomena that should be studied (e.g., fraud).
- ✓ When arbitrary caps distort the natural data distribution.
- ✓ When your model or analysis can handle outliers robustly.

Performing Zero Coding

Zero coding is a technique where outliers, missing values, or invalid data points are replaced with zero (0) in the dataset.

This is often used as a simple way to:

- ✓ Mark or flag problematic values without removing them.
- ✓ Prepare data for models that cannot handle missing or outlier values.
- ✓ Make the data consistent by avoiding NaNs or extreme values.

Why Use Zero Coding?

- ✓ **Simplicity:** Easy to implement and understand.
- ✓ **Model compatibility:** Some machine learning models require no missing or non-numeric values.
- ✓ **Signal representation:** Zero can act as a distinct value indicating missing or outlier cases.
- ✓ **Avoid data loss:** Keeps all records for analysis or modeling.

When to Use Zero Coding?

- ✓ When zero has **meaning or can be interpreted as "no value" or "absent".**
- ✓ When other imputation methods are not practical or necessary.
- ✓ When the dataset or model can tolerate zeros as placeholders.
- ✓ When you want to flag missing or outlier data explicitly.

Important Considerations

- ✓ **Interpretation:** Zero coding can introduce bias if zero is a valid and meaningful value in the dataset (e.g., zero income or zero sales).
- ✓ **Data distribution:** Zero coding can distort data distribution, especially if many values are zeroed.
- ✓ **Alternatives:** Other imputation methods include mean/median substitution, forward filling, or model-based imputation.

Example of Zero Coding for Outliers

Suppose we have the data:
[10, 12, 15, 14, 13, 1000, 11, 13, 15, 9]

Using an IQR method, we identify **1000** as an outlier.

Zero coding the outlier:

- Replace **1000** with 0.

Resulting data:
[10, 12, 15, 14, 13, 0, 11, 13, 15, 9]

```
import pandas as pd

data = pd.Series([10, 12, 15, 14, 13, 1000, 11, 13, 15, 9])

# Define IQR bounds
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Zero code outliers
data_zero_coded = data.apply(lambda x: 0 if (x < lower_bound or x > upper_bound) else x)

print(data_zero_coded)
```

Review Questions

1. Explain why discretization is used in Exploratory Data Analysis (EDA).
2. Differentiate between unsupervised and supervised discretization with examples.
3. Describe the steps in equal-width binning with formula and example.
4. What is desalinization? Justify reasons why EDA used in EDA.
5. For a given data frame apply equal width binning discretization techniques.
   Data [6, 8, 15, 17, 20, 25, 50, 100] and k = 3.
6. Differentiate between equal width binning and equal frequency binning.
7. Describe the core algorithm in equal frequency binning with an example.
8. Illustrate working of decision tree binning with an example.
9. Explain the working of chi-square (x^2) based binning with an example.
10. What are outliers? Explain any four significance of working with outliers.
11. Explain the various interpretations of outliers.
12. List the advantages and disadvantages of equal-width binning.
13. What are the advantages and disadvantages of equal-frequency discretization?
14. Compare equal-width binning and equal-frequency binning.
15. Explain decision tree-based binning with an example.
16. State two advantages and two disadvantages of decision tree binning.
17. Explain Chi-square (χ^2) based binning with steps.
18. What are the advantages and disadvantages of Chi-square binning?
19. Describe entropy-based binning and its application.
20. Define an outlier with an example and explain its significance.
21. Explain different interpretations of outliers.
22. What is trimming of outliers and why is it used?
23. Explain capping and zero coding as methods of handling outliers.

****