Parul® University

# Machine Learning

SUBJECT CODE: 303105353

Parul® University

# CHAPTER - 1

# Machine Learning in Practice

## Agenda

1. Data collection & Preprocessing.
2. Outlier Analysis (Z-Score)
3. Model selection & evaluation
4. Optimization of tuning parameters
5. Visualization of results

## Data collection & Preprocessing.

Data collection and preprocessing are critical steps in the data analysis and machine learning pipeline.

Properly collecting and preparing your data can have a significant impact on the quality and reliability of your results.

## Data collection

1. **Define Objectives**: Clearly define your research or project objectives. What questions do you want to answer or what problems do you want to solve with your data?

2. **Identify Data Sources**: Determine where your data will come from. It might be databases, APIs, web scraping, sensors, surveys, or existing datasets.

3. **Data Gathering**: Collect the data from the identified sources. This can involve writing scripts or programs to automate data retrieval. Ensure that you have the necessary permissions and rights to use the data.

4. **Data Quality Check**: Examine the data for quality issues such as missing values, duplicates, outliers, and inconsistencies. Clean the data as needed to address these issues.

## Data collection

5. **Data Integration**: If your data comes from multiple sources, you may need to integrate it into a single dataset. This may involve data merging, joining, or concatenation.

6. **Data Storage**: Decide on an appropriate storage format and location for your data. Common options include relational databases, NoSQL databases, data lakes, or simple file formats like CSV or JSON.

7. **Data Documentation**: Maintain documentation that describes the data sources, collection methods, and any transformations or cleaning steps performed. This documentation is crucial for reproducibility.

## Data Preprocessing

1. **Handling Missing Data**: Decide how to deal with missing values. You can either remove rows with missing data, impute missing values with statistical methods, or use advanced imputation techniques.

2. **Outlier Detection and Treatment**: Identify and handle outliers that can skew your analysis. You can remove outliers, transform them, or use robust statistical methods.

3. **Feature Selection**: Choose relevant features (columns) that are likely to contribute to your analysis or machine learning model. Feature selection can reduce dimensionality and prevent overfitting.

4. **Feature Engineering:** Create new features that can provide more information or improve model performance. This might involve mathematical transformations, aggregation, or creating categorical variables from continuous data.

5. **Scaling and Normalization**: Scale or normalize your data to ensure that different features have similar scales. Common techniques include min-max scaling and z-score normalization.

## Data Preprocessing

6. **Encoding Categorical Data:** Convert categorical variables into numerical format using techniques like one-hot encoding or label encoding, depending on the nature of the data and the machine learning algorithm you plan to use.

7. **Data Splitting**: Divide your dataset into training, validation, and test sets for machine learning tasks. The training set is used to train models, the validation set is used for hyperparameter tuning, and the test set is reserved for evaluating model performance.

8. **Data Transformation:** Some machine learning algorithms require specific data transformations, such as principal component analysis (PCA) or time series decomposition.

9. **Data Imbalance Handling:** If you're dealing with imbalanced datasets in classification tasks, consider techniques like oversampling, under sampling, or using different evaluation metrics.

10. **Data Visualization:** Visualize your data to gain insights and identify patterns or anomalies. Data visualization tools can help you explore the data's characteristics.

## Data Preprocessing

6. **Encoding Categorical Data:** Convert categorical variables into numerical format using techniques like one-hot encoding or label encoding, depending on the nature of the data and the machine learning algorithm you plan to use.

7. **Data Splitting**: Divide your dataset into training, validation, and test sets for machine learning tasks. The training set is used to train models, the validation set is used for hyperparameter tuning, and the test set is reserved for evaluating model performance.

8. **Data Transformation:** Some machine learning algorithms require specific data transformations, such as principal component analysis (PCA) or time series decomposition.

9. **Data Imbalance Handling:** If you're dealing with imbalanced datasets in classification tasks, consider techniques like oversampling, under sampling, or using different evaluation metrics.

10. **Data Visualization:** Visualize your data to gain insights and identify patterns or anomalies. Data visualization tools can help you explore the data's characteristics.

## Why Data Preprocessing?

- Data in the real world is dirty
  - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., occupation=“ ”
  - noisy: containing errors or outliers
    - e.g., Salary=“-10”
  - inconsistent: containing discrepancies in codes or names
    - e.g., Age=“42” Birthday=“03/07/1997”
    - e.g., Was rating “1,2,3”, now rating “A, B, C”
    - e.g., discrepancy between duplicate records

## Why Is Data Dirty?

- Incomplete data may come from
  - “Not applicable” data value when collected
  - Different considerations between the time when the data was collected and when it is analyzed.
  - Human/hardware/software problems
- Noisy data (incorrect values) may come from
  - Faulty data collection instruments
  - Human or computer error at data entry
  - Errors in data transmission
- Inconsistent data may come from
  - Different data sources
  - Functional dependency violation (e.g., modify some linked data)
- Duplicate records also need data cleaning

## Why Is Data Preprocessing Important?

- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
    - e.g., duplicate or missing data may cause incorrect or even misleading statistics.
  - Data warehouse needs consistent integration of quality data
- Data extraction, cleaning, and transformation comprises the majority of the work of building a data warehouse

## Missing Data

- Data is not always available
  - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
  - not register history or changes of the data
- Missing data may need to be inferred.

Parul® University

DIGITAL LEARNING CONTENT

## How to Handle Missing Data?

- Ignore the tuple: usually done when class label is missing (assuming the tasks in classification—not effective when the percentage of missing values per attribute varies considerably.
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
  - a global constant : e.g., “unknown”, a new class?!
  - the attribute mean
  - the attribute mean for all samples belonging to the same class: smarter
  - the most probable value: inference-based such as Bayesian formula or decision tree

## Noisy Data

- Noise: random error or variance in a measured variable
- Incorrect attribute values may due to
    - faulty data collection instruments
    - data entry problems
    - data transmission problems
    - technology limitation
    - inconsistency in naming convention
- Other data problems which requires data cleaning
    - duplicate records
    - incomplete data
    - inconsistent data

## How to Handle Noisy Data?

- Binning
  - first sort data and partition into (equal-frequency) bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Regression
  - smooth by fitting the data into regression functions
- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human (e.g., deal with possible outliers)

Parul® University

## Data Integration

- Data integration:
  - Combines data from multiple sources into a coherent store
- Schema integration: e.g., A.cust-id ≡ B.cust-#
  - Integrate metadata from different sources
- Entity identification problem:
  - Identify real world entities from multiple data sources, e.g., Bill Clinton = William Clinton
- Detecting and resolving data value conflicts
  - For the same real world entity, attribute values from different sources are different
  - Possible reasons: different representations, different scales, e.g., metric vs. British units

## Handling Redundancy in Data Integration

- Redundant data occur often when integration of multiple databases
  - *Object identification*: The same attribute or object may have different names in different databases
  - *Derivable data:* One attribute may be a "derived" attribute in another table, e.g., annual revenue
- Redundant attributes may be able to be detected by *correlation analysis*
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

## Data Transformation

- Smoothing: remove noise from data
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
- Normalization: scaled to fall within a small, specified range
  - min-max normalization
  - z-score normalization
  - normalization by decimal scaling
- Attribute/feature construction
  - New attributes constructed from the given ones

## Data Transformation: Normalization

- Min-max normalization: to [new_min$_A$, new_max$_A$]

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

  - Ex. Let income range \$12,000 to \$98,000 normalized to [0.0, 1.0]. Then \$73,000 is mapped to $\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716$

- Z-score normalization (μ: mean, σ: standard deviation):

$$v' = \frac{v - \mu_A}{\sigma_A}$$

  - Ex. Let μ = 54,000, σ = 16,000. Then $\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$

- Normalization by decimal scaling

$$v' = \frac{v}{10^j}$$ Where $j$ is the smallest integer such that Max(|v'|) < 1

## Data Reduction Strategies

- Why data Reduction?
  - A database/data warehouse may store terabytes of data
  - Complex data analysis/mining may take a very long time to run on the complete data set.
- Data Reduction
  - Obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the  integrity of the original data.
  - Mining on small data set should be more efficient yet produce the same (or almost the same) analytical results.

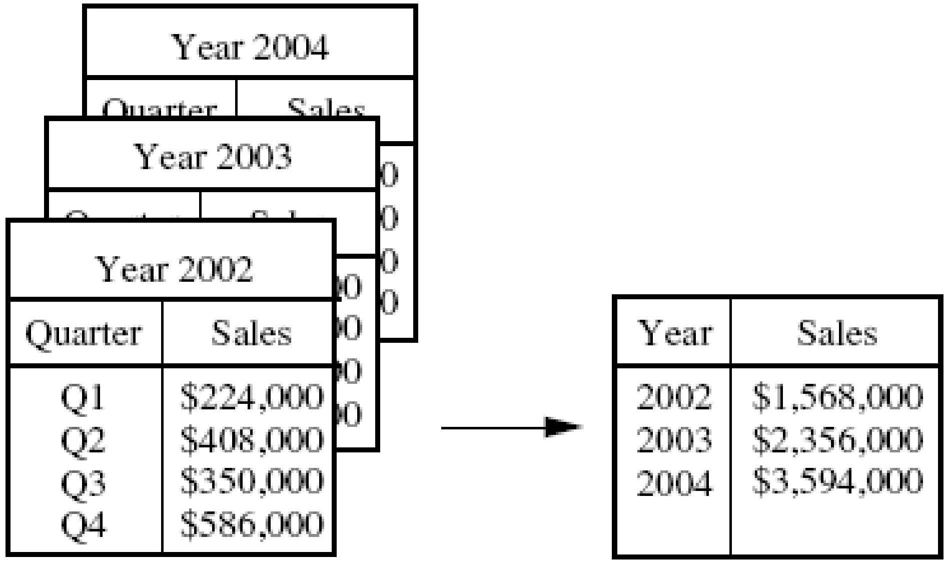
## Data Cube Aggregation

- Stores multidimensional aggregated information.
- Provides fast access to precomputed, summarized data, thereby, benefiting on-line analytical processing as well as data mining.
- The cube created at the lowest level of abstraction is referred to as the base cuboid.
- A cube at the highest level of abstraction is the apex cuboid.
- Data cubes created for varying levels of abstraction are often referred to as cuboids, so that a data cube may instead refer to a lattice of cuboids.

Parul® University

DIGITAL LEARNING CONTENT

## Data Cube Aggregation



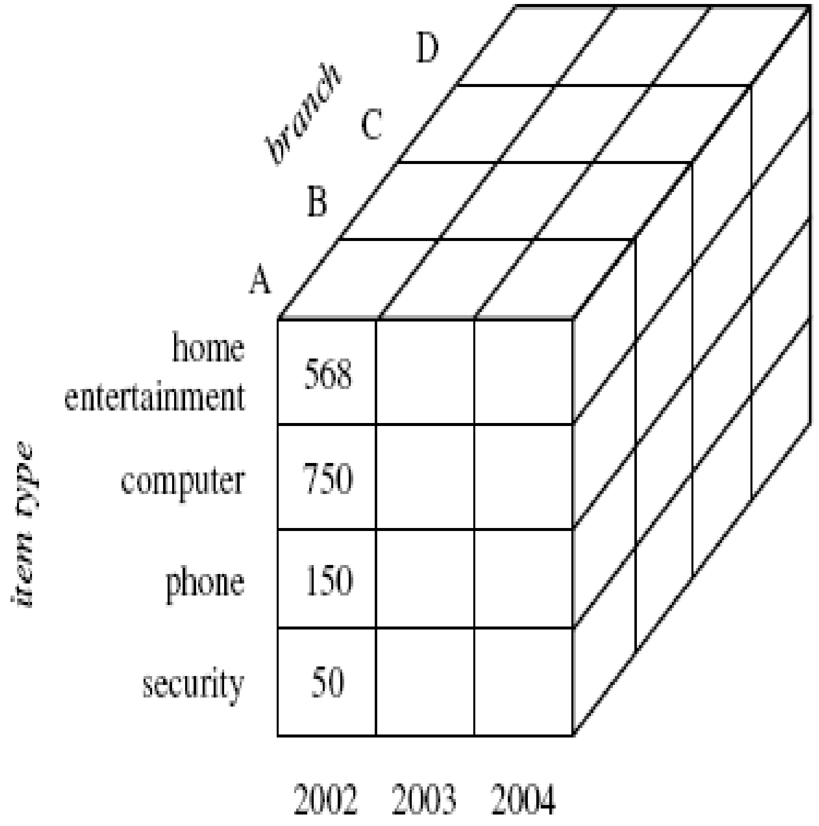
Year 2004

Year 2003

Year 2002

| Quarter | Sales |
|---|---|
| Q1 | $224,000 |
| Q2 | $408,000 |
| Q3 | $350,000 |
| Q4 | $586,000 |

| Year | Sales |
|---|---|
| 2002 | $1,568,000 |
| 2003 | $2,356,000 |
| 2004 | $3,594,000 |



## Attribute Subset Selection

- Feature selection (i.e., attribute subset selection):
  - Select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features
  - reduce # of patterns in the patterns, easier to understand
- Heuristic methods (due to exponential # of choices):
  - Step-wise forward selection
  - Step-wise backward elimination
  - Combining forward selection and backward elimination
  - Decision-tree induction

## Outlier analysis using Z-Score

Outlier analysis using Z-Score, also known as the standard score, is a statistical technique used to identify and deal with outliers in a dataset. Outliers are data points that deviate significantly from the rest of the data and can distort statistical analysis and machine learning models. Z-Score helps you quantify how far each data point is from the mean and provides a threshold for identifying outliers.

## Perform outlier analysis using Z-Score

Calculate the mean (average) and standard deviation of your dataset. These statistics describe the central tendency and the spread of the data, respectively.

**Mean ($\mu$) = ($\Sigma x$) / N**

**Standard Deviation ($\sigma$) = $\sqrt{[\Sigma(x - \mu)^2 / N]}$**

Where:

- $\Sigma$ represents summation (summing over all data points).
- x is each data point.
- N is the number of data points.

## Perform outlier analysis using Z-Score

**Calculate the Z-Score for Each Data Point**:

- For each data point in your dataset, calculate its Z-Score using the formula:

$$\textbf{Z-Score (Z)} = (x - \mu) / \sigma$$

**Z-Score** represents how many standard deviations a data point is away from the mean. A Z-Score of 0 means the data point is exactly at the mean, positive values indicate data points above the mean, and negative values indicate data points below the mean.

## Perform outlier analysis using Z-Score

**Set a Threshold for Identifying Outliers**: Decide on a threshold Z-Score value beyond which data points are considered outliers. A common threshold is, for example, $Z > 2$ or $Z < -2$, which corresponds to data points that are more than two standard deviations away from the mean.

**Identify Outliers**: Data points with Z-Scores exceeding the chosen threshold are considered outliers. You can create a new binary variable to label them as outliers (1) or not (0).

**Handle Outliers:**

1. Remove Outliers: Exclude outlier data points from your analysis, especially if you believe they are erroneous or irrelevant.
2. Transform Data: Apply transformations to mitigate the impact of outliers, such as log transformations or winsorization (capping extreme values).
3. Keep Outliers: In some cases, outliers may be of interest, and you may want to analyze them separately or understand why they exist.

## Perform outlier analysis using Z-Score

**Reanalyze Data:** After handling outliers, you can recompute summary statistics, visualize the data, or build machine learning models with the cleaned dataset.

Z-Score-based outlier analysis is a simple yet effective method for identifying and managing outliers in your data. However, the choice of the Z-Score threshold is somewhat subjective and should be guided by domain knowledge and the specific goals of your analysis. Additionally, be aware that Z-Score-based methods may not work well for datasets with non-normal distributions, and alternative techniques may be more appropriate in such cases.

## Model selection & evaluation

Model selection and evaluation are crucial steps in the process of building and deploying machine learning models. Selecting the right model and assessing its performance correctly are essential for achieving the best results in your machine learning project.

## Model Selection

1. **Define Your Goals:** Clearly articulate the objectives of your machine learning project. Understand what you want to predict or accomplish with your model.

2. **Choose Candidate Models**: Based on your problem type (classification, regression, clustering, etc.) and the nature of your data, select a set of candidate machine learning algorithms. Common choices include linear regression, decision trees, random forests, support vector machines, neural networks, etc.

3. **Feature Selection/Engineering:** Before building and comparing models, carefully select and preprocess your features. Feature engineering may involve creating new features, transforming data, and handling missing or categorical data.

4. **Split the Data**: Divide your dataset into training, validation, and test sets. The training set is used for model training, the validation set for hyperparameter tuning and model selection, and the test set for final model evaluation.

## Model Selection

5. **Train Models:** Train each candidate model using the training data. Tune hyperparameters using the validation set to find the best-performing configuration for each model.

6. **Cross-Validation:** Perform cross-validation on the training data to assess each model's performance more robustly. Common techniques include k-fold cross-validation.

7. **Evaluate Model Complexity:** Consider the trade-off between model complexity and performance. Simpler models are less likely to overfit but may have lower predictive power, while complex models may capture more nuances but are more prone to overfitting.

8. **Select the Best Model:** Based on cross-validation results and your evaluation criteria (e.g., accuracy, precision, recall, F1-score for classification; RMSE, MAE for regression), choose the best-performing model.

## Model Evaluation

1. **Test Data Evaluation:** Once you've selected your best model, evaluate its performance on the test dataset, which it has never seen before. This step provides a realistic estimate of how well your model will perform on unseen data.

2. **Performance Metrics:** Choose appropriate evaluation metrics based on your problem type. For classification, you might use accuracy, precision, recall, F1-score, ROC AUC, etc. For regression, common metrics include RMSE, MAE, R-squared, etc.

3. **Confusion Matrix (Classification):** Analyze the confusion matrix to understand the model's performance regarding true positives, true negatives, false positives, and false negatives. This can help you make informed decisions about trade-offs between precision and recall.

## Model Evaluation

4. **Visualizations:** Create visualizations, such as ROC curves, precision-recall curves, or residual plots, to gain insights into your model's behavior.

5. **Business Impact:** Consider the business or real-world implications of your model's performance. Evaluate whether the model meets the desired objectives and whether it aligns with the project's goals.

6. **Bias and Fairness:** Assess the model for biases, fairness, and ethical concerns. Ensure that it doesn't discriminate against certain groups or exhibit unintended behavior.

7. **Interpretability:** If model interpretability is important, use techniques such as feature importance analysis or model-agnostic interpretability tools to understand how the model makes predictions.

## Model Evaluation

8. **Iterate and Refine:** Depending on the evaluation results, you may need to iterate on the model selection, feature engineering, or data preprocessing steps to improve model performance.

9. **Documentation:** Maintain thorough documentation of the selected model, its hyperparameters, and the evaluation results. This documentation is crucial for reproducibility and future reference.

## Optimization of tuning parameters

Optimizing or tuning hyperparameters is a crucial step in the process of building machine learning models. Hyperparameters are parameters of the model that are not learned from the data but are set before training. Tuning these hyperparameters can significantly impact a model's performance.

## Optimization of tuning parameters

**Define Your Search Space:**
Start by identifying the hyperparameters you want to tune. These may include learning rates, regularization strengths, tree depths, kernel types, etc., depending on the algorithm you're using.

**Choose a Search Method:**
There are two primary methods for hyperparameter tuning:

- •Grid Search: In this method, you specify a set of possible values for each hyperparameter. The algorithm then evaluates all possible combinations, creating a grid of hyperparameter configurations. Grid search is straightforward but can be computationally expensive.

- •Random Search: Random search selects random combinations of hyperparameters from predefined ranges. It's often more efficient than grid search and can find good hyperparameters faster.

## Optimization of tuning parameters

**Set Evaluation Metrics:**

Define the evaluation metric(s) that you want to optimize. This could be accuracy, F1-score, RMSE, etc.

**Cross-Validation:**

Split your training data into multiple subsets for cross-validation. Common choices include k-fold cross-validation, where the data is divided into k subsets, and each model is trained and evaluated on different combinations of these subsets. Cross-validation helps assess how well a set of hyperparameters performs across different data partitions, reducing the risk of overfitting.

## Optimization of tuning parameters

**Hyperparameter Tuning:**
Apply the chosen search method (grid search or random search) to find the best hyperparameters. For each combination of hyperparameters:

- •Train the model on the training data.
- •Evaluate the model's performance using cross-validation and the chosen evaluation metric(s).
- •Record the evaluation metric(s) for that combination.

**Select the Best Hyperparameters:**
After trying multiple hyperparameter combinations, choose the set that results in the best performance on your chosen evaluation metric(s). This set is considered the optimized set of hyperparameters.

## Optimization of tuning parameters

**Test Set Evaluation:** Assess the model with the optimized hyperparameters on a separate test dataset that it has not seen during the hyperparameter tuning process. This gives you an estimate of how well your model will perform on new, unseen data.

**Refinement:** Depending on the results, you may need to iterate on the optimization process, fine-tuning the hyperparameters further or even revisiting your initial choices.

**Documentation:** Document the optimized hyperparameters and the performance metrics associated with them. This documentation is essential for reproducibility and model deployment.

**Deployment and Monitoring:** Deploy your model with the optimized hyperparameters in a production environment. Monitor its performance over time and be prepared to re-tune the hyperparameters periodically as the data distribution evolves or the model's requirements change.

## Visualization of results

Visualizing the results of your data analysis or machine learning model can provide valuable insights, help you communicate your findings effectively, and aid in decision-making. The choice of visualization techniques depends on the type of data and the specific goals of your analysis.

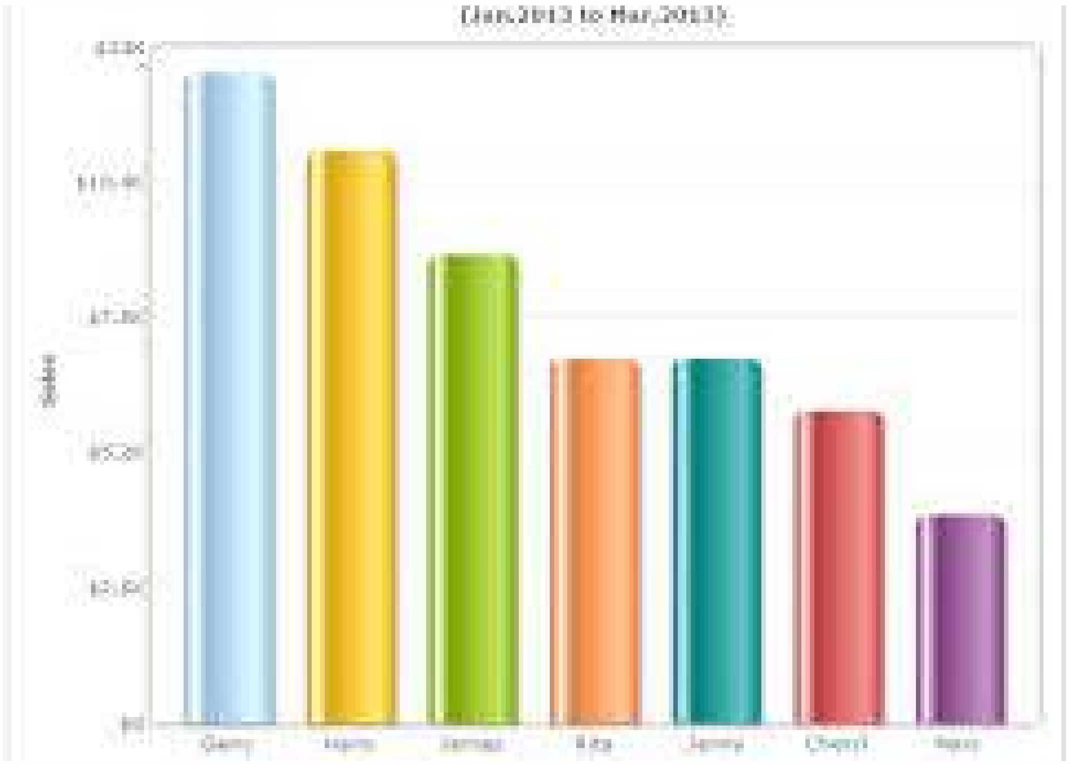
Parul® University

DIGITAL LEARNING CONTENT

## Common ways to visualize

### Bar Charts and Histograms

1. Use bar charts to display categorical data and compare the frequency or distribution of different categories.
2. Histograms are useful for visualizing the distribution of continuous data. They divide the data into bins and display the frequency of data points in each bin.



Parul® University

DIGITAL LEARNING CONTENT

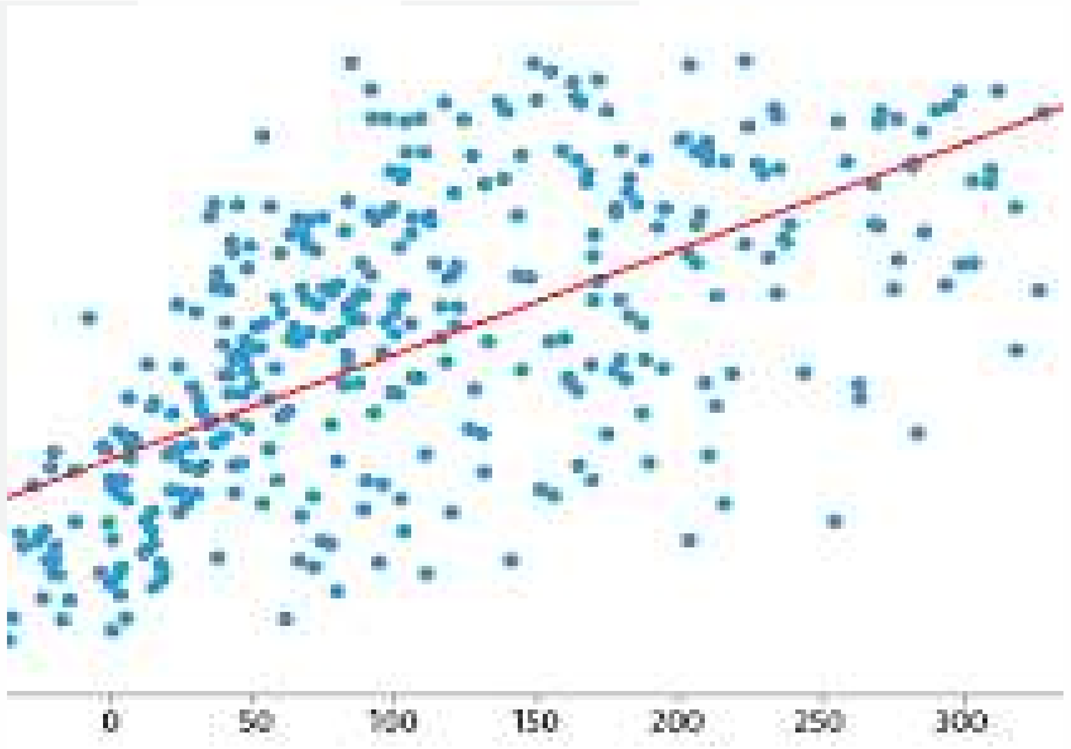
## Common ways to visualize

**Line Charts**

Line charts are ideal for showing trends over time or across ordered categories. They are commonly used for time series data or to visualize the relationship between two continuous variables.

**Scatter Plots**

Scatter plots are effective for visualizing the relationship between two continuous variables. Each point on the plot represents a data point, making it easy to identify patterns or outliers.



## Common ways to visualize

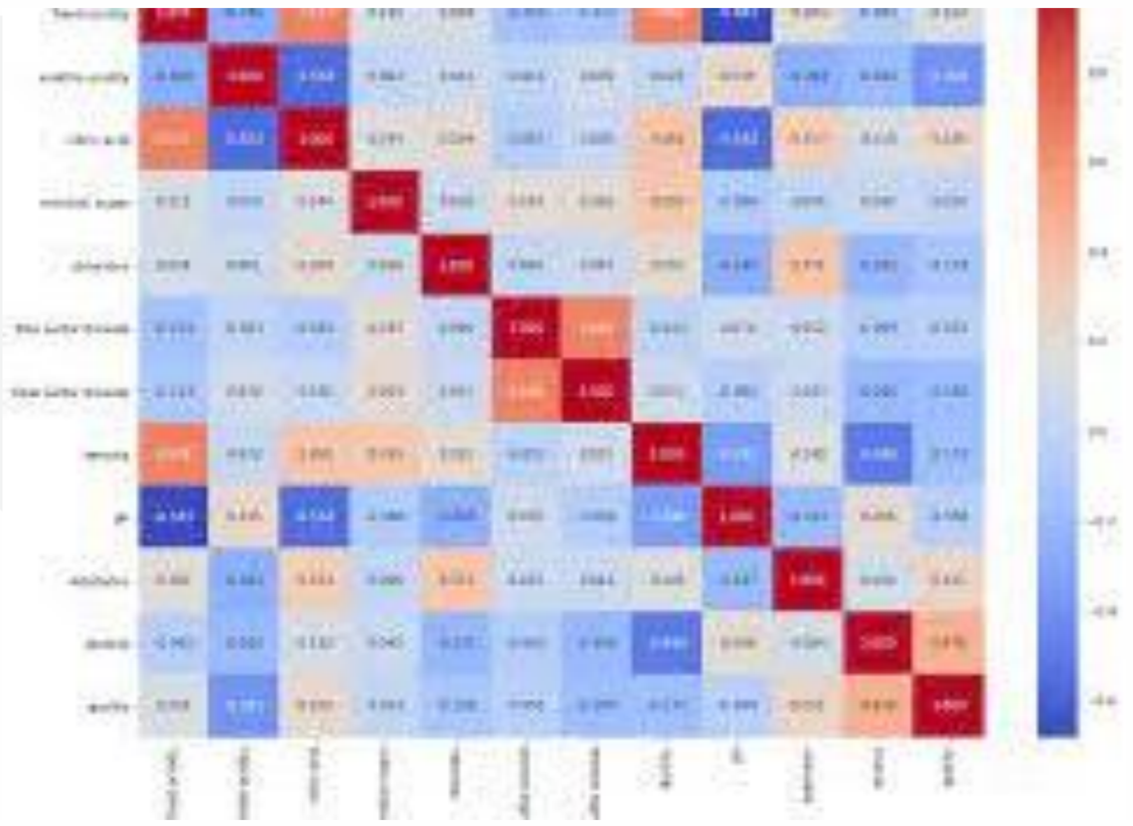
**Box Plots and Violin Plots**

Box plots summarize the distribution of a dataset, showing median, quartiles, and potential outliers.

•Violin plots combine a box plot with a kernel density estimation, providing a more detailed view of the data's distribution.

**Heatmaps:**

Heatmaps are used to represent data in a tabular format where colors indicate the values of individual cells. They are commonly used for correlation matrices or to visualize data in a 2D grid.



Parul® University

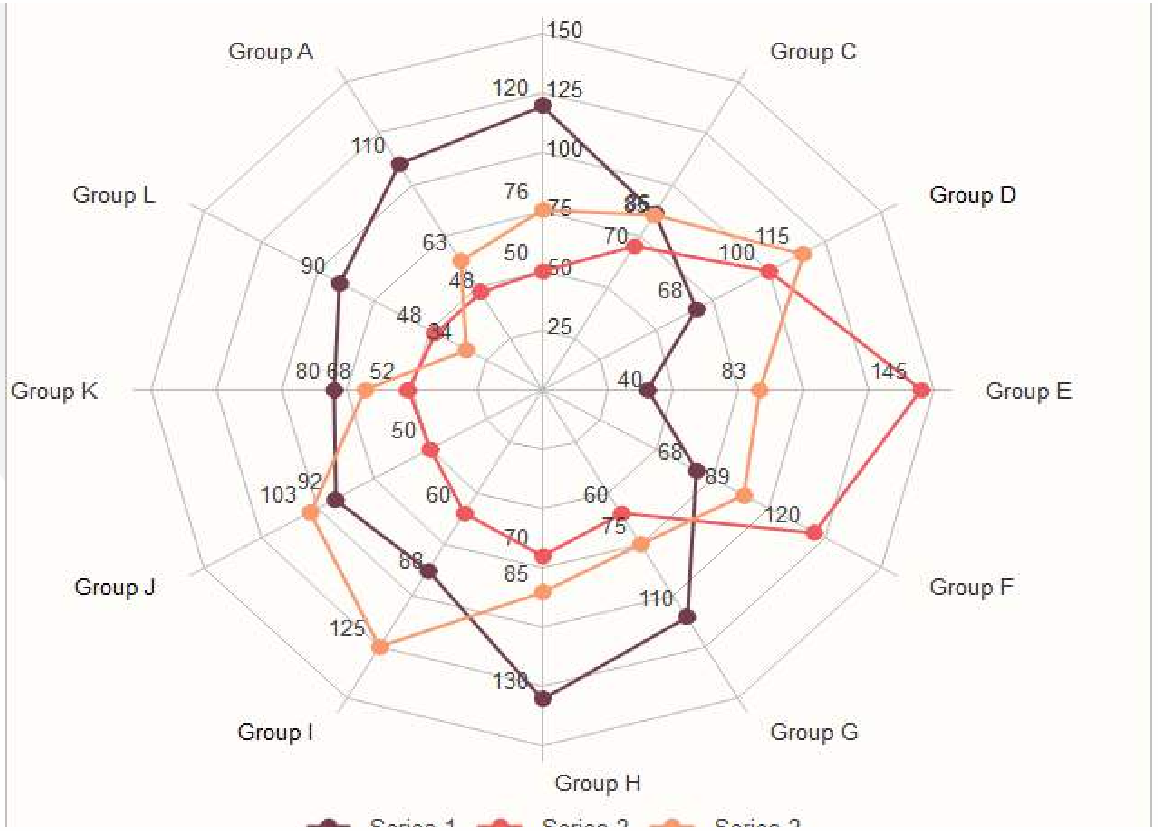
DIGITAL LEARNING CONTENT

## Common ways to visualize

**Pie Charts:**
Pie charts display parts of a whole. They are suitable for showing the composition of a dataset when you have a small number of categories.

**Area Charts:**
Area charts are similar to line charts but are filled in with color, making it easier to visualize the cumulative effect of values over time.

**Radar Charts (Spider Charts):**
Radar charts are useful for displaying multivariate data on a two-dimensional chart with multiple axes. They are suitable for comparing items across multiple categories.

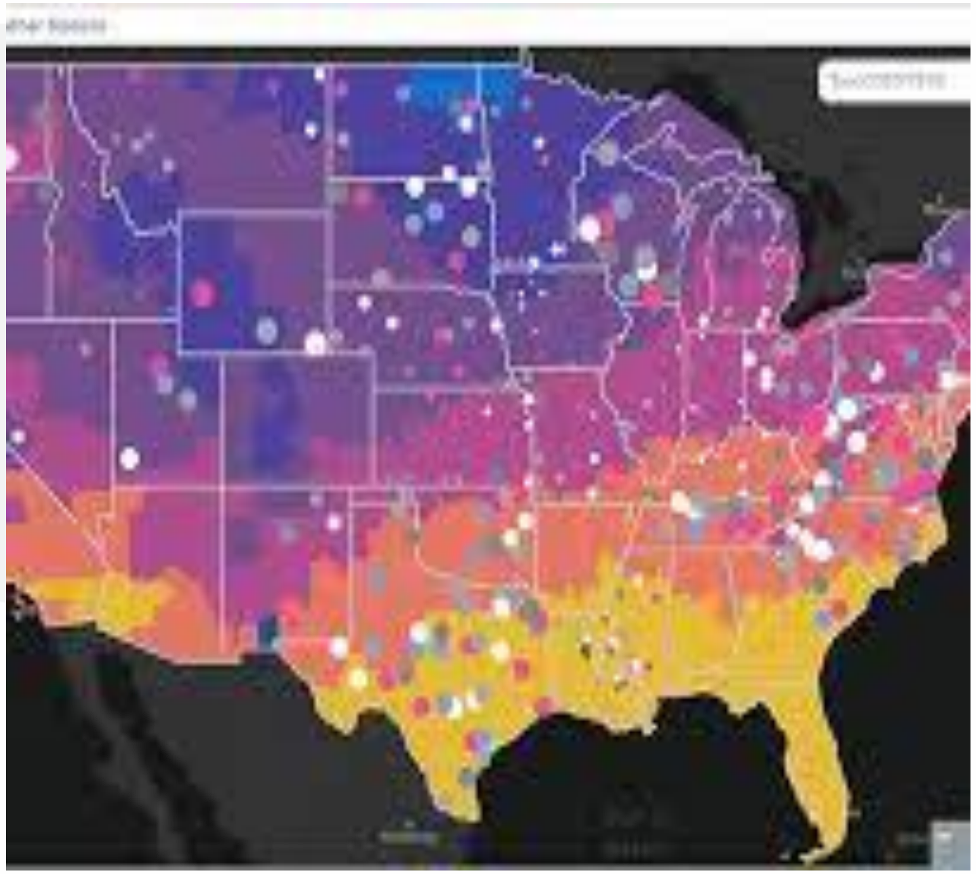


# Common ways to visualize

**Bubble Charts**
Bubble charts add a third dimension to scatter plots by varying the size of data points based on a third variable. They are useful when you need to represent three dimensions of data.

**Sankey Diagrams**
Sankey diagrams are used to visualize the flow of resources or quantities between different entities. They are often used in process analysis or to depict hierarchical structures.

**Choropleth Maps**
Choropleth maps use color-coding to represent data by geographic regions. They are useful for showing regional patterns or variations.

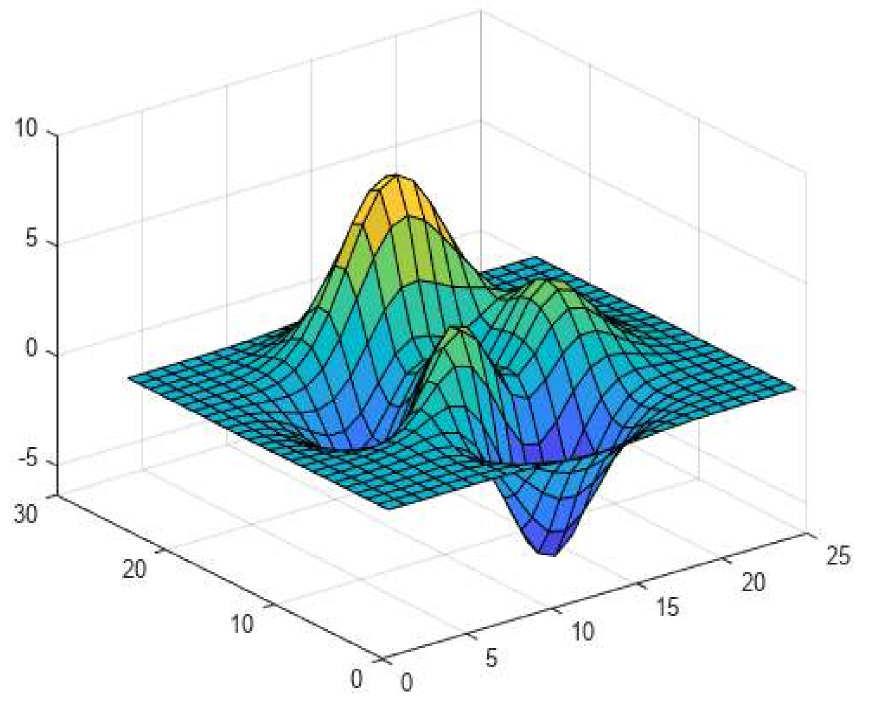


## Common ways to visualize

**3D Plots**
3D plots can be used when you need to visualize data in three dimensions. They are suitable for situations where two continuous variables are dependent on a third variable.

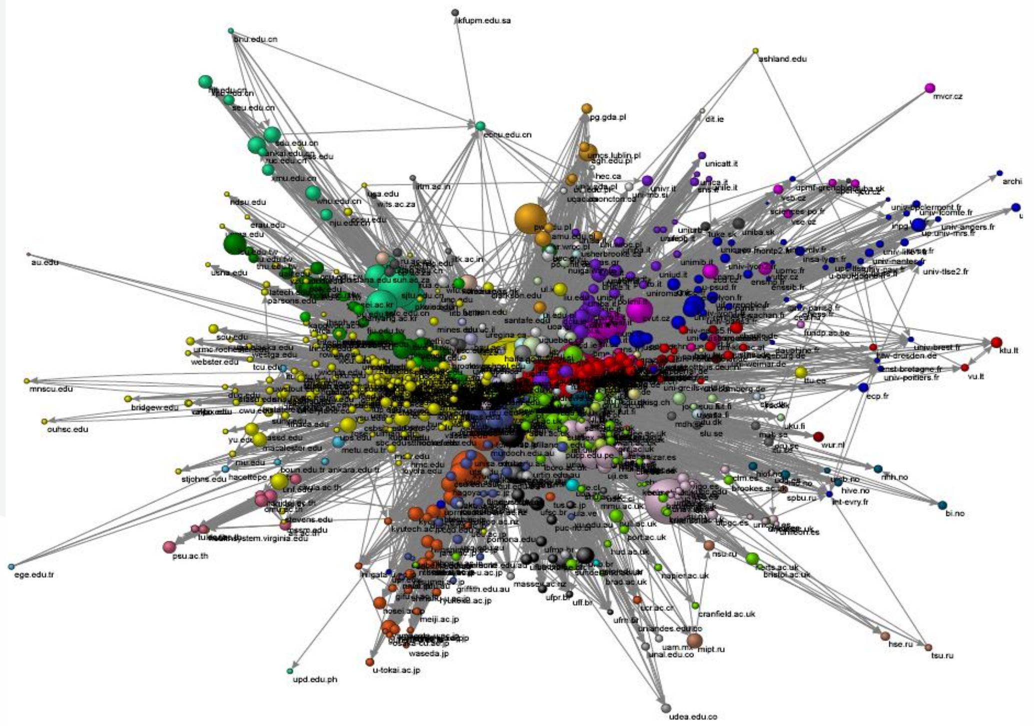
**Interactive Dashboards:**
Create interactive dashboards using tools like Tableau, Power BI, or Plotly. These allow users to explore data and results dynamically.

**Word Clouds and Text Visualization:**
Word clouds are used to visualize word frequencies in text data. Other techniques like sentiment analysis or topic modeling can also be used to visualize and interpret text data.



## Common ways to visualize

**Network Graphs:**
Network graphs are useful for visualizing relationships between entities, such as social networks, co-authorship networks, or hierarchical structures.



**Error Bars:**
When presenting statistical results, use error bars to indicate variability or uncertainty in your measurements.

Thank You