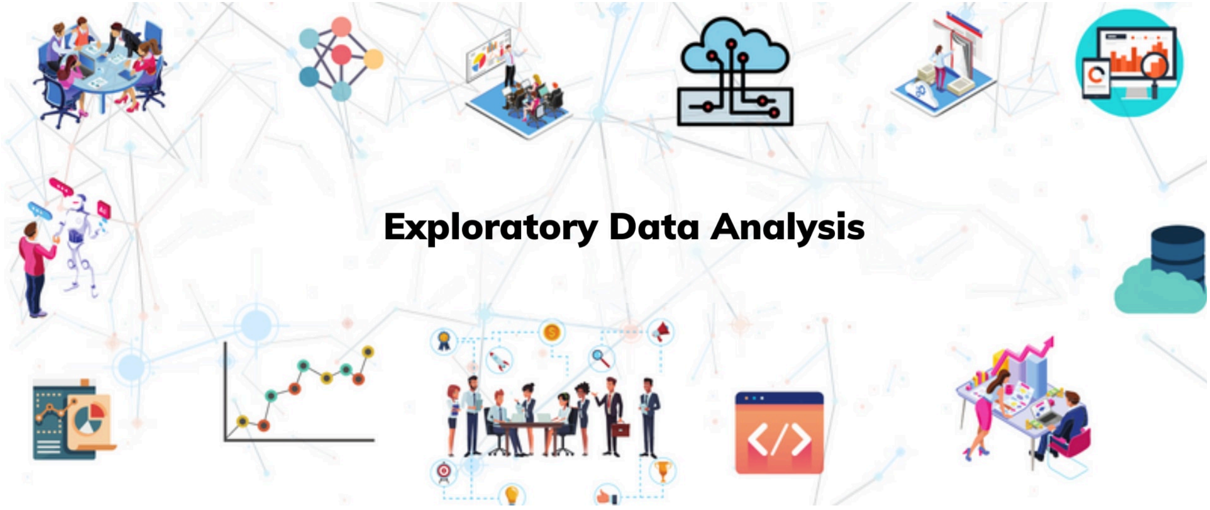


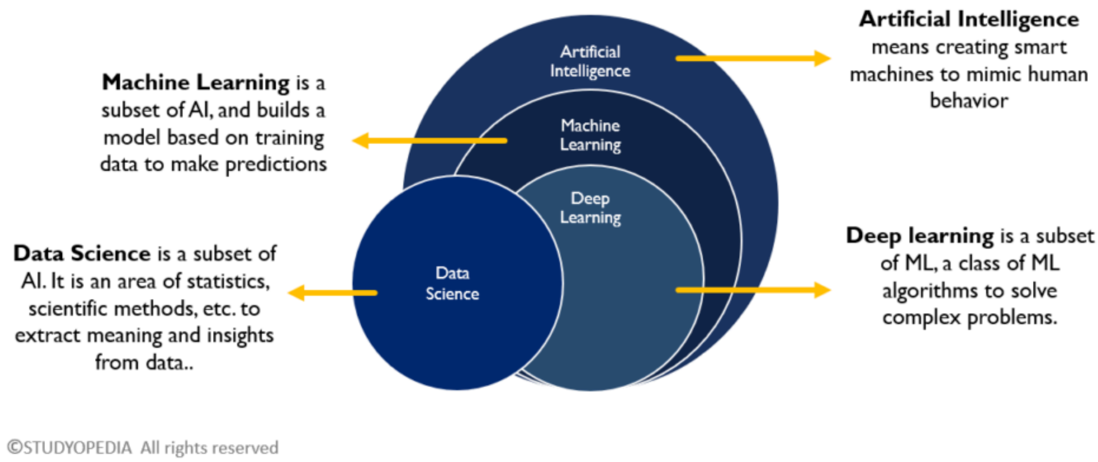
MOOC 1: Exploratory Data Analysis for Machine Learning - Study Notes

🔗 **View Original Notion Document**: https://www.notion.so/MOOC-1-Exploratory-Data-Analysis-for-Machine-Learning-Study-Notes-27b2d55efe0c809b97e6d9d5c2c30ef0?source=copy_link

---

MODULE 1: AI, ML & DEEP LEARNING

Core Definitions



ARTIFICIAL INTELLIGENCE (AI)

- Any program that can sense, reason, act and adapt
- Simulation of intelligent behavior in computers

MACHINE LEARNING (ML)

- **Subset of AI** - Learns from data, improves over time
- Not explicitly programmed, but learns patterns from data

DEEP LEARNING (DL)

- **Subset of ML** - Uses multi-layered neural networks
- Automatically learns features, no manual feature engineering needed

AI History Timeline

- **1950s**: Turing Test developed, AI term officially coined
- **1960s-70s**: **First AI Winter** - Disappointment with early results
- **1980s**: Expert Systems boom → **Second AI Winter** follows
- **1990s-2000s**: ML breakthrough successes (Google PageRank, speech recognition)
- **2012+**: Deep Learning revolution begins (AlexNet breakthrough)

Real-world Applications

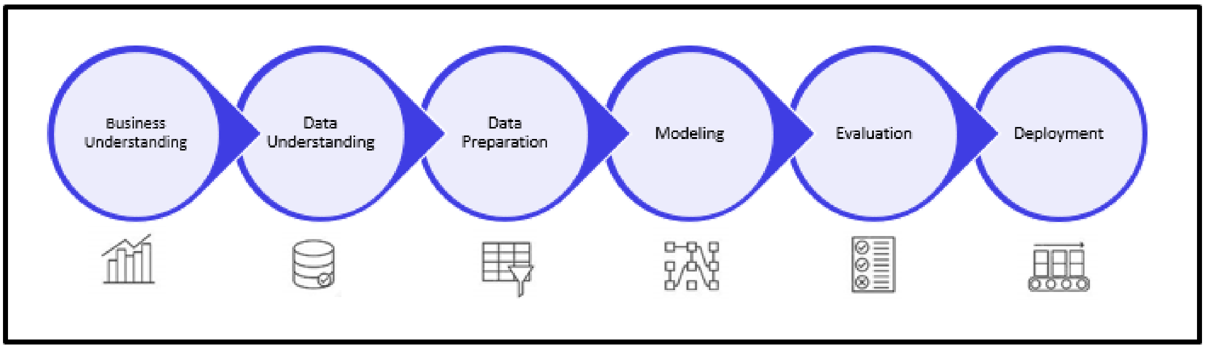
- **Computer Vision**: Face recognition, autonomous vehicles
- **Natural Language Processing**: Machine translation, sentiment analysis
- **Healthcare**: Medical imaging analysis, drug discovery
- **Business Intelligence**: Fraud detection, recommendation engines

---

MODULE 2: DATA RETRIEVAL & CLEANING

Machine Learning Workflow

Problem Statement → Data Collection → Data Exploration & Preprocessing → Modeling → Validation → Deployment



Detailed Workflow Steps:

1. **Problem Statement**: Define the business problem and success metrics
2. **Data Collection**: Gather relevant data from various sources
3. **Data Exploration & Preprocessing**: Clean, explore, and prepare data
4. **Modeling**: Select and train appropriate ML algorithms
5. **Validation**: Evaluate model performance and generalization
6. **Deployment**: Implement model in production environment

Data Sources Overview

Type	Format	Python Library	Use Case
CSV	Comma-separated	`pd.read_csv()`	Tabular data
JSON	JavaScript Object	`pd.read_json()`	APIs, NoSQL
SQL	Structured queries	`sqlite3`, `sqlalchemy`	Relational DB
NoSQL	Document-based	`pymongo`	MongoDB

Data Cleaning Strategies

GARBAGE IN = GARBAGE OUT

Data quality determines model performance!

Missing Values Handling

- **Remove**: Lose data but no bias introduced
- **Impute**: Replace with mean/median - adds uncertainty
- **Mask**: Treat missing as separate category

```
# Missing values handling examples
df.dropna()              # Remove rows with missing values
df.fillna(df.mean())       # Impute with mean
```

```
df.fillna(df.median())      # Impute with median
df.fillna('Missing')        # Mask as separate category
```

Outlier Detection Methods

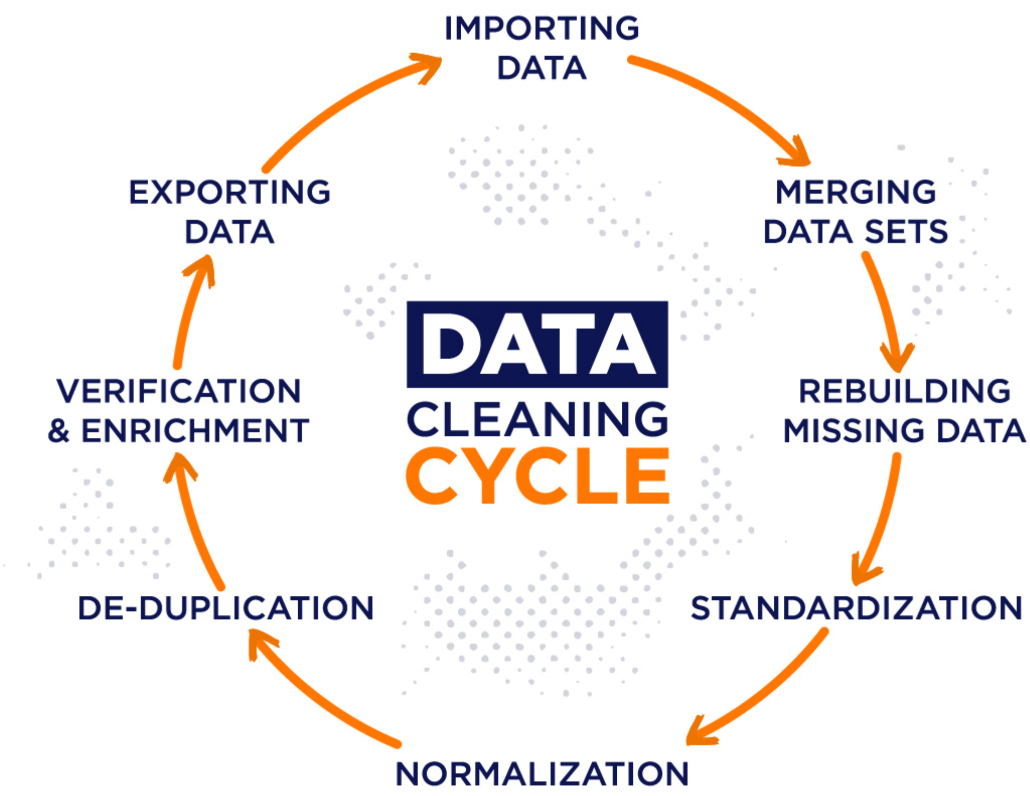
1. **Visual Methods**: Histograms, Box plots, Scatter plots
2. **Statistical Methods**:
   - **IQR Method**: Q3 + 1.5×IQR (upper bound)
   - **Z-Score**: |z| > 3 → outlier
   - **Residuals**: Standardized, Deleted, Studentized

```
# Outlier detection using IQR method
Q1 = df['column'].quantile(0.25)
Q3 = df['column'].quantile(0.75)
IQR = Q3 - Q1
outliers = df[(df['column'] < (Q1 - 1.5 * IQR)) | (df['column'] > (Q3 + 1.5 * IQR))]

# Z-score method
z_scores = np.abs((df['column'] - df['column'].mean()) / df['column'].std())
outliers = df[z_scores > 3]
```

Duplicate Data Handling

- **Investigate first**: Distinguish real duplicates vs. valid repeated observations
- **Remove carefully**: Ensure no important information is lost



MODULE 3: EXPLORATORY DATA ANALYSIS & FEATURE ENGINEERING

EDA - First Conversation with Data

EDA Goals

1. **Understand** data structure & quality
2. **Identify** patterns & relationships
3. **Detect** issues needing cleaning
4. **Guide** modeling decisions

Statistical Summary Functions

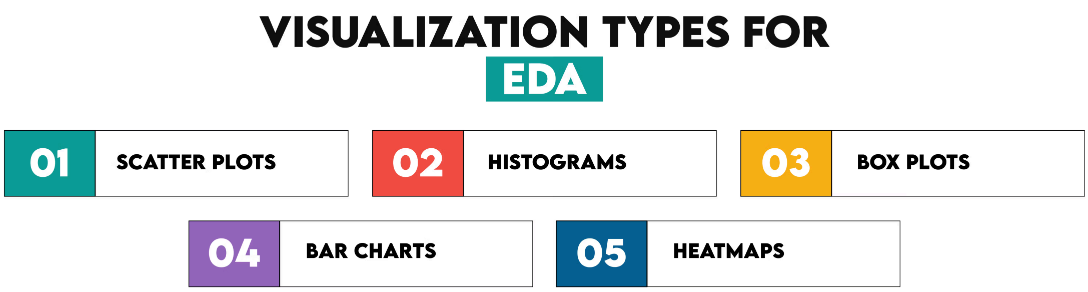
```
# Core EDA functions
df.describe()         # Summary statistics
```

```
df.info()           # Data types & nulls
df.corr()           # Correlations
df.value_counts()   # Category frequencies

# Basic visualizations
df['column'].hist()   # Histogram
df.boxplot()          # Box plot
sns.pairplot(df)      # Pair plot for relationships
```

Visualization Libraries

- **Matplotlib**: Foundation library - maximum flexibility
- **Seaborn**: Beautiful statistical plots - `pairplot()`, `heatmap()`
- **Pandas**: Quick plotting - `df.plot()` for rapid insights



Feature Engineering Techniques

Variable Transformations

- **Log Transform**: Reduces positive skew → normal distribution
- **Polynomial Features**: x, x^2, x^3 to capture non-linear relationships
- **Box-Cox**: Automatically finds optimal transformation

```
# Variable transformations
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from scipy.stats import boxcox

# Log transformation
df['log_column'] = np.log1p(df['column'])  # log(1+x) to handle zeros

# Polynomial features
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)
```

Encoding Categorical Variables

ENCODING METHODS

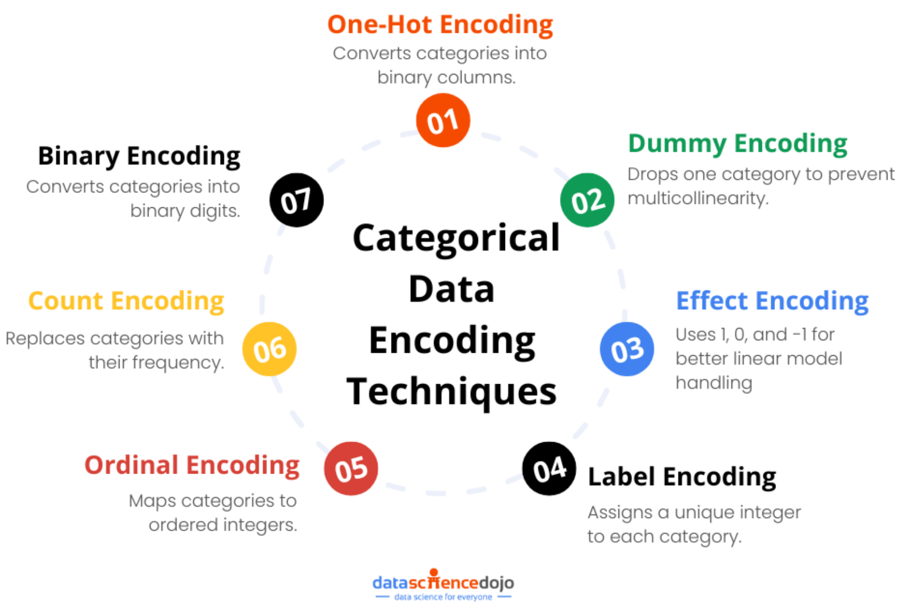
- **Binary Encoding**: True/False → 0/1
- **One-Hot Encoding**: Multiple categories → multiple binary columns
- **Ordinal Encoding**: Low/Medium/High → 1/2/3 (preserves order)

```
# Categorical encoding examples
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder

# One-hot encoding
df_encoded = pd.get_dummies(df, columns=['category_column'])

# Binary/Label encoding
le = LabelEncoder()
df['encoded_column'] = le.fit_transform(df['category_column'])

# Ordinal encoding (preserves order)
ordinal_encoder = OrdinalEncoder(categories=[['Low', 'Medium', 'High']])
df['ordinal_encoded'] = ordinal_encoder.fit_transform(df[['ordered_column']])
```



Feature Scaling Comparison

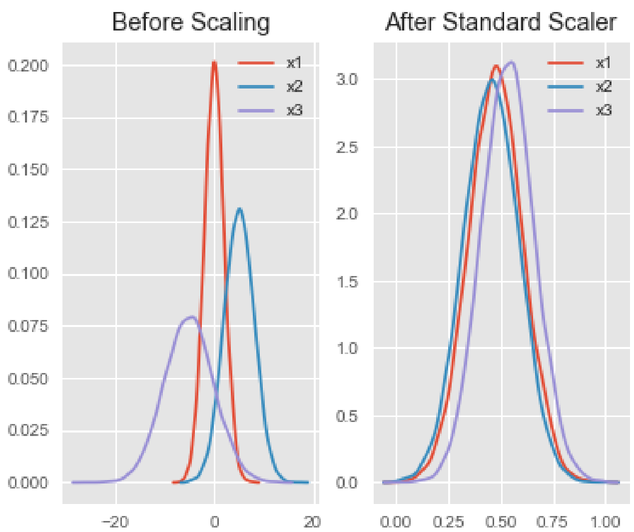
Method	Formula	Range	Outlier Effect
Standard Scaling	$(x - \mu)/\sigma$	Unbounded	Moderate
Min-Max Scaling	$(x - min)/(max - min)$	[0,1]	**High Risk**
Robust Scaling	$(x - median)/IQR$	Unbounded	**Low Risk**

```
# Feature scaling examples
from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler

# Standard scaling
scaler = StandardScaler()
X_standard = scaler.fit_transform(X)

# Min-Max scaling
min_max_scaler = MinMaxScaler()
X_minmax = min_max_scaler.fit_transform(X)

# Robust scaling (recommended for outliers)
robust_scaler = RobustScaler()
X_robust = robust_scaler.fit_transform(X)
```



MODULE 4: STATISTICAL INFERENCE & HYPOTHESIS TESTING

Key Statistical Concepts

ESTIMATION vs INFERENCE

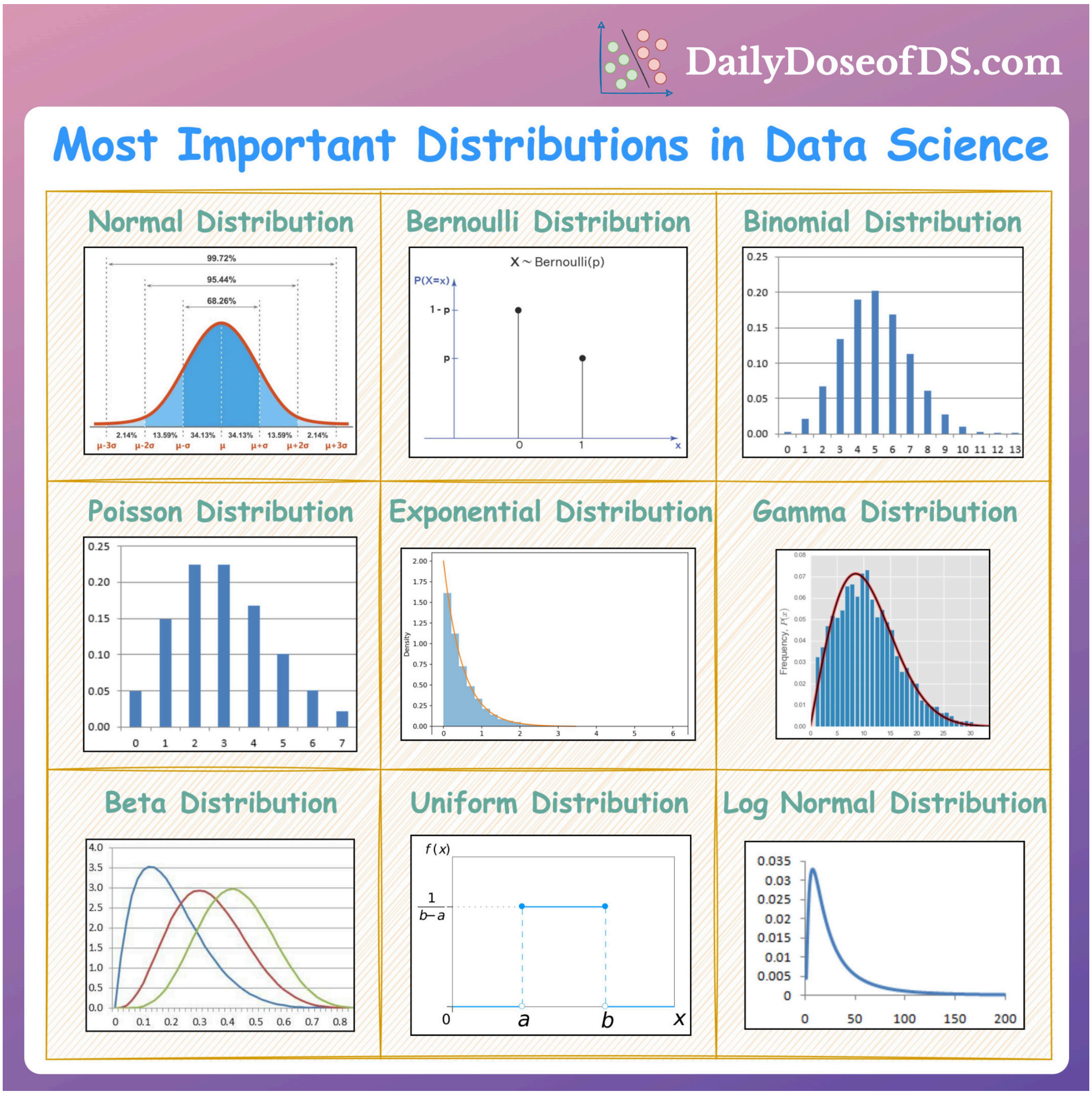
- **Estimation**: Point estimate (e.g., mean = 25)
- **Inference**: Interval + uncertainty (e.g., 95% CI: 20-30)

Parametric vs Non-parametric Approaches

- **Parametric**: Assumes specific distribution (e.g., Normal) + finite parameters
- **Non-parametric**: Distribution-free, relies on actual data structure

Common Statistical Distributions

Distribution	Use Case	Parameters	Example
Normal	Heights, test scores	μ (mean), σ (std)	Most natural phenomena
Binomial	Success/Failure trials	n (trials), p (probability)	Coin flips, A/B testing
Poisson	Count events in time	λ (rate parameter)	Website visits per hour
Exponential	Time between events	λ (rate parameter)	Time to next customer



Hypothesis Testing Framework

HYPOTHESIS SETUP

- **H_0 (Null Hypothesis)**: No effect exists, status quo
- **H_1 (Alternative Hypothesis)**: Effect exists, change from status quo
- **α (Significance Level)**: Type I error tolerance (typically 0.05)

Error Types in Hypothesis Testing

- **Type I Error (α)**: False Positive - Incorrectly reject true H_0
- **Type II Error (β)**: False Negative - Fail to reject false H_0

- **Statistical Power**: $1-\beta$ = Probability of correctly rejecting false H_0

P-value Interpretation

> **P-value = Probability of observing data this extreme if H_0 is true**

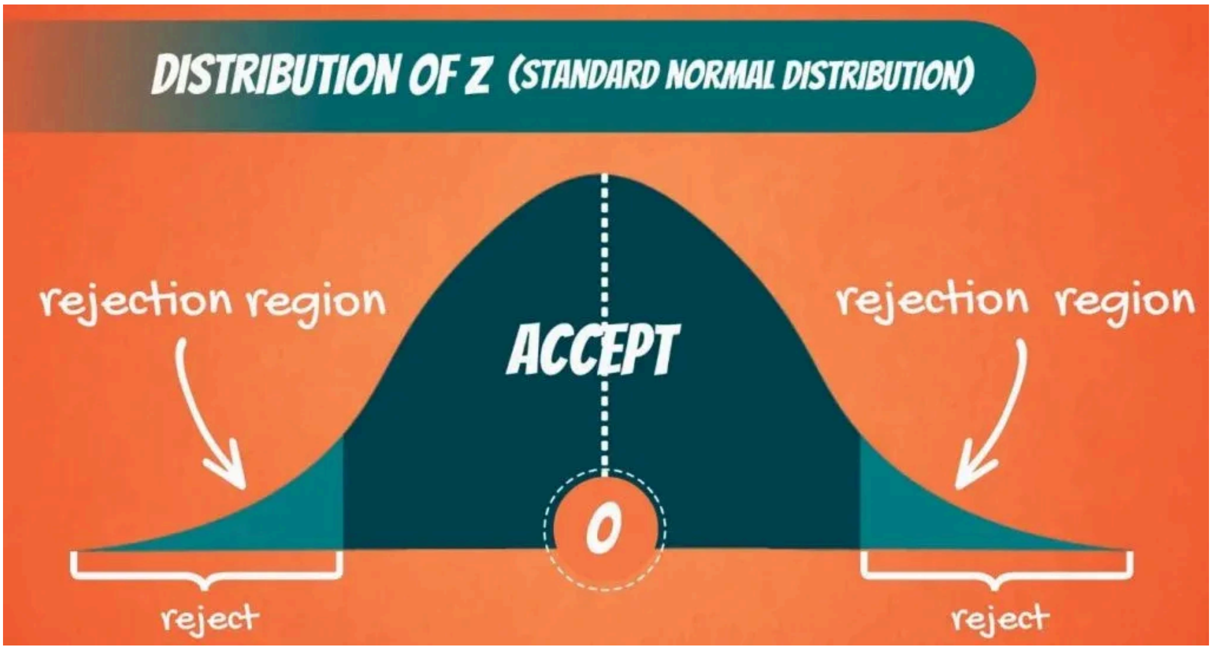
Decision Rules:

- **$p < 0.05$**: Reject H_0 (statistically significant result)
- **$p \geq 0.05$**: Fail to reject H_0 (insufficient evidence)

MULTIPLE TESTING PROBLEM

With n independent tests: $P(\geq 1 \text{ Type I error}) \approx n \times \alpha$

Solution: Bonferroni Correction → Use α/n for each test

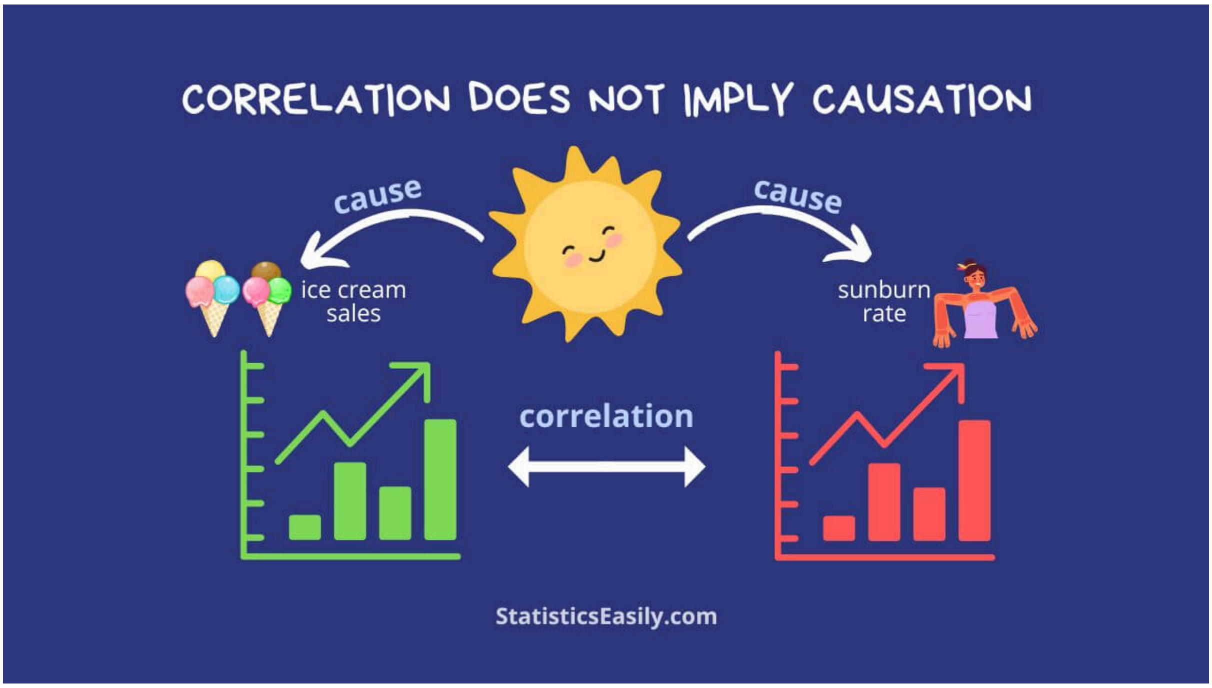


Correlation vs Causation

CORRELATION ≠ CAUSATION

Four Possible Relationships:

1. **X → Y**: X directly causes Y
2. **Y → X**: Y directly causes X (reverse causation)
3. **Z → X,Y**: Confounding variable Z causes both
4. **Spurious**: Random correlation, no real relationship



Business Examples

- **Incorrect Interpretation**: More customer service calls → Lower satisfaction
- **Correct Interpretation**: Lower satisfaction → More customer service calls

---

BUSINESS APPLICATION: CUSTOMER CHURN PREDICTION

CHURN PREDICTION WORKFLOW

Target Variable: Customer leaves (1) or stays (0)

Feature Variables: Tenure, purchase history, demographics, usage patterns

Model Output: Probability score (0.0 to 1.0)

Hypothesis Testing Example:

- **H_0**: Customer tenure has no effect on churn probability
- **H_1**: Longer tenure significantly reduces churn probability

---

PYTHON TOOLS & LIBRARIES

Essential Libraries

```
import pandas as pd          # Data manipulation and analysis
import numpy as np           # Numerical computing and arrays
import matplotlib.pyplot as plt  # Basic plotting and visualization
import seaborn as sns         # Statistical data visualization
import scipy.stats as stats   # Statistical functions and tests
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

Key Functions by Category

```
# Data Loading and I/O
pd.read_csv(), pd.read_json(), pd.read_sql()

# Exploratory Data Analysis
df.describe(), df.info(), df.corr()
sns.pairplot(), sns.heatmap(), sns.boxplot()

# Data Preprocessing
StandardScaler(), MinMaxScaler(), LabelEncoder()
pd.get_dummies()  # One-hot encoding

# Statistical Testing
scipy.stats.ttest_1samp()  # One-sample t-test
scipy.stats.pearsonr()     # Pearson correlation test
scipy.stats.chi2_contingency()  # Chi-square test
```