CHAPTER 4

PRESCRIPTIVE ANALYTICS

INTRODUCTION TO PRESCRIPTIVE ANALYTICS

Detailed View of Prescriptive Analytics

Prescriptive Analytics is a sophisticated form of data analysis that not only forecasts future outcomes but **recommends specific actions** to achieve desired results. It integrates data, mathematical models, machine learning, and business rules to **suggest optimal decisions** in complex scenarios.

Prescriptive Analytics recommends the **optimal course of action** to achieve **specific goals**. It answers the question **"What should we do?"** by leveraging insights from **descriptive** and **predictive analytics**, employing **optimization techniques** and **simulations**, while considering **constraints** and **objectives** to provide **actionable recommendations**.

- Recommends the **optimal course of action**.
- Aims to answer: **"What should we do?"**
- Leverages descriptive and predictive analytics.
- Uses optimization techniques and simulations.
- Considers constraints and objectives.

🔁 Analytics Maturity Ladder

Level	Question Answered	Example
Descriptive Analytics	What happened?	Sales dropped by 10% last quarter.
Diagnostic Analytics	Why did it happen?	Due to lower footfall and weather.
Predictive Analytics	What might happen next?	Sales may drop again this quarter.
Prescriptive Analytics	**What should we do about it?**	Offer discounts, change inventory, etc.

🧩 Core Components of Prescriptive Analytics

Component	Description
Data	Cleaned and enriched historical and real-time data.
Predictive Models	Machine learning or statistical models to predict outcomes.
Optimization Engine	Mathematical models (e.g., linear programming) to find best decisions under constraints.
Business Rules/Logic	Domain-specific rules (e.g., budgets, policies) to guide feasible recommendations.
Scenario Analysis	"What-if" modeling to explore consequences of different actions.
Automation Systems	Integration with applications to automatically implement recommendations (e.g., ERP, CRM).

🛠️ Techniques Used in Prescriptive Analytics

Technique	Purpose	Example Use Case
Linear Programming (LP)	Optimize a linear objective function with constraints.	Minimize delivery cost while satisfying demand.
Integer Programming (IP)	Same as LP but with integer solutions.	Staff scheduling.
Simulation	Test different decisions in a simulated environment.	Emergency response planning.
Markov Decision Processes	Model decisions in stochastic processes.	Predict customer churn and recommend actions.
Heuristics/Metaheuristics	Solve complex optimization problems with approximation methods.	Vehicle routing problems.
Reinforcement Learning	Learn optimal policies through trial and error with feedback.	Dynamic pricing systems.

🏢 Business Application Scenarios

Industry	Prescriptive Use Case
Retail	Personalized promotions to maximize ROI.
Healthcare	Treatment plan optimization to improve patient outcomes.
Banking	Recommend loan approval decisions with risk constraints.
Logistics	Route optimization for delivery to reduce fuel costs and time.
Telecom	Suggest customer retention strategies based on usage patterns.
Energy	Smart grid optimization for energy load distribution.

📈 Benefits of Prescriptive Analytics

- **Action-Oriented Insights**: Not just what will happen, but what to do about it.
- **Informed Decision-Making**: Supports complex decision processes with data and models.
- **Proactive Strategy**: Enables businesses to respond *before* problems occur.
- **Resource Optimization**: Efficient use of capital, human, and technical resources.
- **Competitive Advantage**: Creates agility and automation in strategy execution.

⚠️ Challenges in Prescriptive Analytics

Challenge	Description
Data Quality	Inaccurate or incomplete data can lead to poor recommendations.
Model Complexity	Sophisticated models may be hard to interpret or explain.
Real-Time Processing	Need for fast computation in dynamic environments.
Integration	Difficulty in integrating with existing systems.
User Trust	Stakeholders may be reluctant to trust automated decisions.

🛡️ Best Practices

- Validate models with domain experts.

- Continuously monitor and update recommendations.
- Combine with human judgment for high-stakes decisions.
- Start with a pilot project before full-scale implementation.
- Build explainability into models to improve trust.

📌 Use Case in Business with Real-time Example (Customer Retention)

✅ Customer Retention Optimization Example

Scenario:
A subscription-based video streaming company (e.g., StreamBox) is experiencing a drop in active users and wants to understand why — and how to retain more customers.

🔹 Step 1: Descriptive Analytics

Analyzes usage data over the last 12 months:

- 25% of users dropped off after 3 months of usage.
- Peak usage occurs during weekends and at night.
- Genre-wise: Comedy and Drama are most watched; Documentaries have the least engagement.

Tool: SQL reports, Tableau dashboards

🔹 Step 2: Predictive Analytics

Using classification and clustering models to predict churn:

- Customers with less than 5 hours/week of usage and no profile personalization are **70% likely to churn**.
- Segment identified: **Low-engagement users aged 18–25** in Tier-2 cities are at higher risk.

Tool: Logistic regression, Random Forest, k-means clustering

🔹 Step 3: Prescriptive Analytics

Now, the company asks:
"What actions should we take to reduce churn and increase customer engagement?"

Prescriptive Model Recommends:

- Launch a **personalized weekly content suggestion engine**.
- Offer a **loyalty bonus** (1-month free) to users likely to churn.
- Run a **targeted campaign** for Tier-2 cities with mobile data-friendly streaming.
- Introduce **gamification** (badges, streaks) to increase usage time.

Tools Used:

- Optimization algorithms (budget allocation for campaigns)
- Recommendation systems
- Rule-based engines (e.g., if watch-time < 3h, trigger retention flow)

🔁 **Result:**

After implementation:

- Churn rate **reduced by 22%** in the target segment.
- Average user engagement **increased by 15%**.
- Revenue from recurring subscriptions **increased by ₹12 lakhs in 3 months**.

TYPES OF PRESCRIPTIVE ANALYTICS

1. **OPTIMIZATION**

1. Optimization

- **Sample Simplified Formula (Linear Programming Objective):** Maximize $Z = c_1x_1 + c_2x_2$ (where Z is the objective like profit, c represents profit per unit, and x represents the quantity of each variable).
- **Focus:** Identifies the mathematically *best* solution by finding the optimal values for decision variables within given constraints.
- **Application:** Used for resource allocation, production planning, and pricing strategies to maximize gains or minimize losses.

Step	Description
1. **Define decision variables**	What are you trying to decide or control? (e.g., quantity to produce, price to set)
2. **Formulate the objective function**	What are you trying to maximize or minimize? (e.g., profit, cost, time)
3. **Identify constraints**	What limits or rules apply? (e.g., budget, resources, capacity)
4. **Choose a solving method**	Linear → Simplex; Nonlinear → GRG, Newton, Evolutionary
5. **Solve & interpret**	Analyze the optimal solution and interpret decision values

A company is launching a new product and wants to **maximize profit**. However, **demand decreases as price increases**, and production cost is fixed.

Step	Description	Example (with data)
1. Define decision variables	What are you trying to decide or control?	Let **P** = price of the product
2. Formulate the objective function	What are you trying to maximize or minimize?	Maximize **profit**, where: Demand = $1000 - 40P + 1.5P^2$ Profit = (P × Demand) – $10,000 (fixed cost)
3. Identify constraints	What limits or rules apply?	P must be between $10 and $30 (price range allowed) Demand should not exceed 1500 units (capacity constraint)
4. Choose a solving method	Linear → Simplex Nonlinear → GRG, Newton, Evolutionary	This is **nonlinear** (due to squared term), so use **GRG Nonlinear** or **Evolutionary Solver** in Excel
5. Solve & interpret	Analyze the optimal solution and interpret decision values	Solver shows that the **optimal price is $21.45**, resulting in **maximum profit of $13,870**

✅ Business Case: Ad Budget Allocation Optimization

🎯 Goal:

A company wants to **maximize sales** by allocating a limited digital ad budget across three channels: Google, Instagram, and LinkedIn.

📊 Data Available:

Channel	Expected Sales per ₹1 Invested	Max Budget Limit
Google	₹8	₹40,000
Instagram	₹5	₹30,000
LinkedIn	₹6	₹20,000

- Total available ad budget: ₹70,000

🔢 Step 1: Define Decision Variables

Let:

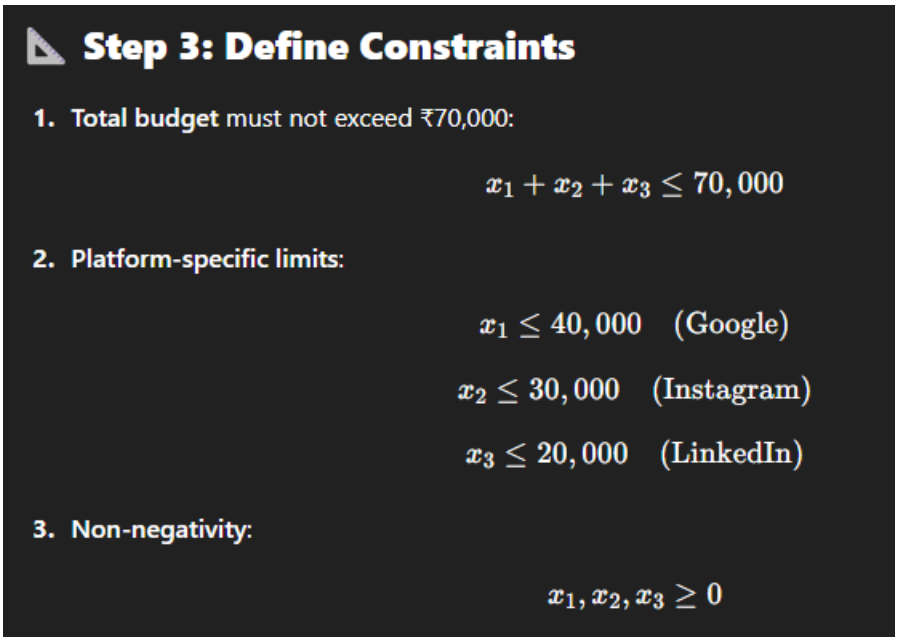
- x1x_1x1 = amount to spend on **Google Ads**
- x2x_2x2 = amount to spend on **Instagram Ads**
- x3x_3x3 = amount to spend on **LinkedIn Ads**

🧮 Step 2: Define Objective Function

Maximize total expected sales:

$$\text{Maximize } Z = 8x_1 + 5x_2 + 6x_3$$

📐 Step 3: Define Constraints



⚙️ Step 4: Solve Using Linear Programming

You can solve this using tools like:

- **Excel Solver**
- **Python (PuLP, SciPy)**
- **R (lpSolve)**

✅ Excel Solver Steps:

1. Set Objective: Maximize total sales function.
2. Set Decision Variables: x1,x2,x3
3. Add constraints as described above.
4. Use Simplex LP method and solve.

📈 Step 5: Interpretation of Solution

Let's say Solver gives this result:

- x1=40,000 (Google – Maxed out)
- x2=30,000 (Instagram – Maxed out)
- x3=0 (LinkedIn – Not used)

Total budget used = ₹70,000
Total sales = Z=8(40,000)+5(30,000)+6(0)=₹320,000+₹150,000=₹470, 000

🔍 Conclusion

- Best strategy is to invest fully in **Google** and **Instagram** only.
- **LinkedIn** provides a lower ROI, so it's skipped.
- Business achieves maximum sales for the given budget.

---

2. SIMULATION

2. Simulation

- Sample Simplified Formula (Monte Carlo Simulation Output): `Expected Outcome = Σ (Probability of Scenario$_i$ * Outcome of Scenario$_i$)` across many iterations.
- Focus: Creates virtual models to test "what-if" scenarios and understand the potential range of outcomes from different decisions.
- Application: Useful for risk analysis, supply chain modeling, and financial forecasting to evaluate the robustness of strategies under uncertainty.

Step	Description
1. Define decision alternatives	List the different strategies or actions available to the decision-maker.
2. Identify uncertain outcomes	Specify possible events or results following each decision, along with their probabilities.
3. Assign payoffs to each outcome	Estimate the payoff (monetary or utility-based) for each combination of decision and outcome.
4. Calculate expected values (EV)	Multiply each outcome's payoff by its probability and sum them to get expected values for each decision path.
5. Select decision with highest EV	Choose the alternative that offers the best expected value or aligns with the decision-maker's preferences or risk profile.

A startup is deciding whether to **launch a new mobile app** now or **test it in a small market first**. The outcome of a full launch depends on how the market responds, which is uncertain.

They want to **maximize expected profit**, but also account for **risk and uncertainty**.

🔁 Solution 1: Simulation-Based Prescriptive Analytics

Step	Description	Example (App Launch)
1. Define uncertain variables	What values are random or unknown?	App success rate = 65%, Profit = $500K Failure = –$300K
2. Build the simulation model	Use formulas to link inputs to outcome	`=IF(RAND()<0.65, 500000, -300000)` in Excel
3. Repeat simulation many times	Generate 10,000 random outcomes	Run a Monte Carlo simulation to get 10,000 profit values
4. Analyze distribution of outcomes	Look at average, risk, extremes	Mean profit = $200K 20% chance of loss Worst case = –$300K
5. Make decision under uncertainty	Decide based on risk tolerance	App launch is profitable on average, but risky. If risk-averse → consider test market first.

✅ Business Case: Inventory Planning via Simulation

🎯 Goal:

A retail company wants to **decide optimal stock levels** for a best-selling product (e.g., smartphones) during a 30-day festive sale period, considering uncertain **daily demand**.

📊 Business Context:

- Selling Price: ₹20,000
- Cost Price: ₹14,000
- Unsold stock incurs a ₹1,000 holding cost per unit
- Lost demand incurs ₹2,000 per unit (lost goodwill/sales)

🔢 Step 1: Understand the Uncertainty

- Based on historical data, **daily demand** follows a **normal distribution**:
  - Mean demand = 100 units/day
  - Standard deviation = 20 units/day

🧮 Step 2: Simulation Objective

Test multiple **inventory levels** (e.g., 90, 100, 110, ..., 140 units/day) and simulate **30 days of demand** for each level to:

🔎 Find the stock level that maximizes expected profit

⚙️ Step 3: Model Assumptions

For each day:

- If **demand ≤ stock**, all units are sold → revenue = units sold × ₹20,000
- If **demand > stock**, some demand is lost → loss = (demand - stock) × ₹2,000
- If **stock > demand**, extra units incur holding cost = (stock - demand) × ₹1,000
- Profit = Revenue – Cost – Holding cost – Lost sales cost

🧪 Step 4: Run Monte Carlo Simulation

Simulate for each stock level:

- Generate 1,000 different 30-day demand scenarios using random sampling from the normal distribution.
- For each scenario, compute total profit for that inventory level.
- Average the profits across all simulations.

Tools: Python (numpy, matplotlib, simpy), Excel with RAND(), @Risk add-on, or R

📈 Step 5: Analyze Results

Let's say average simulated profits were:

Daily Stock Level	Avg Profit Over 30 Days

90	₹1,200,000
100	₹1,275,000
110	₹1,310,000 ✅ (Best)
120	₹1,280,000
130	₹1,230,000

📌 Conclusion

- The **optimal inventory level is 110 units/day**, based on simulated demand scenarios.
- This minimizes **lost sales and holding costs** while maximizing profit.
- Decision is **data-driven**, not based on assumptions.

🧰 Tools You Can Use

- **Python:** NumPy, Pandas, Matplotlib for simulation
- **Excel:** Use NORM.INV(RAND(), mean, stdev) to generate demand
- **@Risk or Simul8:** Drag-and-drop business simulation tools

---

3. DECISION RULES AND HEURISTICS

3. Decision Rules and Heuristics

- Sample Simplified Formula (If-Then Rule): `IF (Inventory Level < Threshold) THEN Order More Stock`.
- Focus: Employs pre-defined business logic or expert-derived rules to recommend actions based on specific conditions.
- Application: Often used for automated decision-making in areas like credit scoring, fraud detection, and customer service routing where speed and consistency are important.

Step	Description
1. Define decision alternatives	List the different strategies or actions available to the decision-maker.
2. Identify uncertain outcomes	Specify possible events or results following each decision, along with their probabilities.
3. Assign payoffs to each outcome	Estimate the payoff (monetary or utility-based) for each combination of decision and outcome.
4. Calculate expected values (EV)	Multiply each outcome's payoff by its probability and sum them to get expected values for each decision path.
5. Select decision with highest EV	Choose the alternative that offers the best expected value or aligns with the decision-maker's preferences or risk profile.

A logistics company is deciding whether to:

- **Invest in electric delivery vans now**
- **Wait one year and reassess**
- **Continue using diesel vans**

The company is concerned about uncertain fuel prices and environmental regulations that may impact operating costs.

Step	Description	Example (with Data)
1. Define decision alternatives	List the choices available to the company	1. Buy electric vans now 2. Wait and decide after 1 year 3. Keep diesel vans
2. Identify uncertain outcomes	Define future events and their probabilities	Fuel prices may **increase (70%)** or **stay flat (30%)** Government may **impose carbon tax (60%)** or not (40%)
3. Assign payoffs to each outcome	Estimate the financial result for each scenario	**Electric vans**: Cost = $500K; Annual savings = $120K × 10 = $1.2M → Net = $700K **Diesel vans**: No investment; potential fuel/tax loss = –$50K/year → Net = –$500K over 10 years **Wait**: Invest in Year 2, cost = $500K, savings = $100K × 9 = $900K, delay cost = –$50K → Net = **$350K**
4. Calculate expected values (EV)	Compute weighted average outcomes	EVs already calculated in payoffs: **EV (Electric Now)** = $700K **EV (Wait)** = $900K – $500K – $50K = **$350K** **EV (Diesel)** = –$500K
5. Select decision with highest EV	Choose the best strategy based on EV or risk preference	**Electric Vans Now** has the highest expected value (**$700K**) and is the recommended choice unless risk aversion or flexibility is prioritized

✅ Business Case: Credit Card Fraud Detection Using Decision Rules

🎯 Goal:

A bank wants to **automatically flag potentially fraudulent transactions** based on a set of rules derived from past fraud patterns.

📊 Context & Challenge

- Huge volume of daily transactions
- Full predictive models are expensive to deploy in real-time
- A **rule-based system** offers a quick and explainable solution

---

🔢 Step 1: Define Business Heuristics (If–Then Rules)

Based on historical fraud analysis, frauds often have these patterns:

Rule ID	Business Heuristic
R1	If transaction amount > ₹50,000 **AND** international location → Flag as fraud
R2	If >3 transactions from different countries within 1 hour → Flag as fraud
R3	If card used in 2 cities >500 km apart in 2 hours → Flag as fraud
R4	If transaction at odd hours (1–5 AM) **AND** not usual behavior → Flag as fraud

These heuristics are built from **domain expert knowledge + data patterns**.

⚙️ Step 2: Implement a Rule-Based Engine

The rules are applied sequentially to each transaction. For a new transaction:

Example Transaction:

- Amount: ₹60,000
- Location: USA
- Time: 3:30 AM
- Previous txn: 10 minutes ago in India

Apply Rules:

- R1: ✓ Yes → Amount > ₹50k & international → Flag
- R2: ✓ Yes → >3 countries in 1 hour → Flag
- R3: ✓ Yes → Distance too high → Flag
- R4: ✓ Yes → Odd time + user rarely shops at night → Flag

➡️ **Transaction is flagged by all 4 rules** → Strong fraud indicator.

🧮 Step 3: Combine Rule Outcomes

You can assign **weights or confidence levels**:

Rule	Match?	Weight	Score
R1	Yes	0.3	0.3

R2	Yes	0.2	0.2
R3	Yes	0.25	0.25
R4	Yes	0.25	0.25
Total Score			**1.0** ✅ Fraud

Set a **threshold** (e.g., score > 0.6 = fraud)

📈 Step 4: Continuous Refinement

- Track **false positives/negatives**
- Periodically **update rules or add new ones**
- Optionally, use these rules as **features** in a predictive model later

---

📌 Conclusion

- Simple rules **automate fast fraud detection**
- Transparent, auditable, explainable
- Works well as a **first layer defense** in real-time systems

---

🛠 Tools You Can Use

- **Rule Engines:** Drools (Java), Business Rule Frameworks (Python custom), Rete algorithm
- **No-code Tools:** Power Automate (Microsoft), decision trees in RapidMiner
- **CRM Integration:** Integrate into Salesforce or banking backend

---

GENERAL OVERVIEW OF THE TECHNIQUES:

📘 1. OPTIMIZATION

Aspect	Description
Definition	A mathematical technique used to **maximize or minimize** an objective (e.g., profit, cost, time) under **specific constraints**.
Goal	Find the **best possible solution** from a set of feasible options.

Key Formula	Linear Programming (LP): Maximize $Z = c_1x_1 + c_2x_2 + \dots + c_nx_n$ Subject to: $a_1x_1 + a_2x_2 + \dots \leq b$
Tools	Excel Solver, Python (PuLP, SciPy), Gurobi, LINGO
Business Example	Maximize delivery efficiency by assigning vehicles to routes based on fuel, time, and load constraints.
Use Case	Supply chain optimization, pricing strategies, production planning

---

💉 2. SIMULATION

Aspect	Description
Definition	A method that **models uncertainty** by simulating thousands of possible scenarios to **understand outcomes and risks**.
Goal	Analyze what might happen under different random inputs.
Key Concept	Monte Carlo Simulation – generate random values from probability distributions to simulate outcomes.
Tools	Python (numpy, random), Excel + RAND(), @Risk, Arena, Simul8
Business Example	Simulate daily sales demand during a festival to decide how much stock to keep.
Use Case	Risk analysis, inventory planning, financial forecasting, queuing systems

---

🧠 3. DECISION RULES (Heuristics)

Aspect	Description
Definition	A set of **if–then rules** or heuristics used to make decisions quickly based on expert knowledge or past patterns.
Goal	Automate and speed up decisions when mathematical models are too slow or complex.
Key Formula	No mathematical formula – relies on **business logic** or simple scoring rules. Example: IF purchase > ₹50,000 AND outside city → Flag as risk

Tools	Business rule engines (Drools, EasyRules), Python if logic, CRM systems
Business Example	Approve/reject loan applications based on age, income, credit score rules.
Use Case	Fraud detection, eligibility checks, alerts, compliance decisions

---

✅ Comparison Summary

Feature	Optimization	Simulation	Decision Rules
Focus	Best outcome under constraints	Uncertainty & scenario testing	Fast rules-based decisions
Based on	Math models (equations)	Randomness / probability	If–Then logic / expert rules
Outcome	Optimal solution	Distribution of possible outcomes	Clear go/no-go actions
Best for	Complex planning, resource allocation	Risky/variable environments	Real-time decisions, alerts

---

Prescriptive Modeling: Detailed Overview

📘 What is Prescriptive Modeling?

Prescriptive modeling is a branch of analytics that focuses on providing **recommendations** on **what actions to take** to achieve desired outcomes. It goes beyond descriptive and predictive analytics by **suggesting the best course of action** based on past data, predictive insights, and defined business goals.

While **predictive analytics** tells you **what will happen**, **prescriptive analytics** tells you **what should happen** and suggests steps to optimize results. It involves using optimization, simulation, and heuristic methods to create actionable solutions.

🔧 Key Components of Prescriptive Modeling

1. **Optimization Models**
    - **Goal**: Maximize or minimize an objective (e.g., cost, revenue, efficiency).

- **Method**: Solve a mathematical problem that maximizes or minimizes a goal subject to constraints (e.g., Linear Programming, Integer Programming).
- **Example**: Maximizing profit in production planning by determining the optimal number of units to produce of each product, given resource constraints.

2. **Simulation Models**
   - **Goal**: Simulate real-world scenarios with uncertainty to understand potential outcomes and assess risk.
   - **Method**: Use Monte Carlo simulations, queuing models, or agent-based models to simulate different scenarios and assess their impact on the business.
   - **Example**: Simulating customer demand and stock levels to determine the best stock replenishment strategy that minimizes costs while meeting demand.
3. **Decision Rules & Heuristics**
   - **Goal**: Apply simple decision rules to quickly solve complex problems.
   - **Method**: Use **if-then logic** to handle decisions where full optimization may not be practical or possible in real time.
   - **Example**: Creating a decision rule to approve or reject loan applications based on credit score, income, and past payment behavior.
4. **Machine Learning (AI) Models**
   - **Goal**: Leverage data-driven models to recommend actions based on learned patterns from historical data.
   - **Method**: Use supervised or unsupervised learning algorithms (e.g., decision trees, reinforcement learning) to continuously optimize decision-making.
   - **Example**: A recommendation engine for retail stores suggesting products to customers based on their browsing history and preferences.

🔢 Mathematical Formulation in Prescriptive Modeling

The goal in prescriptive modeling is typically to **maximize or minimize an objective** while adhering to various constraints. These constraints are often formulated as linear or non-linear equations.

1. **Optimization Problem Example (Linear Programming)**

1. **Optimization Problem Example (Linear Programming)**

Objective Function:

$$\text{Maximize } Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$$

Where:

- Z is the objective (e.g., profit, cost).
- $x_1, x_2, \ldots, x_n$ are the decision variables (e.g., quantity of products to produce).
- $c_1, c_2, \ldots, c_n$ are the coefficients (profit per unit).

Subject to constraints:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2$$

Where:

- a_{ij} represents the coefficients for each constraint.
- $b_1, b_2, \ldots$ represent the available resources.

Constraints could include limitations on budget, time, or resources.

Steps in Prescriptive Modeling

1. **Problem Definition**:
    - Define the objective: What do you want to achieve? (e.g., maximize profit, minimize cost, optimize efficiency)
    - Identify decision variables: What can be controlled? (e.g., amount of resources, production quantities)
    - Define constraints: What limits or restrictions exist? (e.g., resource limits, market demands)
2. **Data Collection and Analysis**:
    - Gather data from relevant sources: historical data, market analysis, system logs, etc.
    - Analyze the data for patterns, trends, and relationships that can inform decision-making.
3. **Model Selection**:

- **Optimization**: Linear, Integer, or Non-linear programming models.
- **Simulation**: Monte Carlo simulations, queuing theory, agent-based modeling.
- **Heuristics**: Rule-based systems, decision trees, expert judgment.

4. **Modeling and Solution**:
   - Build the model using appropriate tools (e.g., Python, Excel, optimization software).
   - Solve the model to get optimal recommendations (e.g., best product quantities to produce, best pricing strategy).
5. **Implementation**:
   - Put the model's recommendations into action, integrate them into business processes.
   - Ensure proper systems and workflows are in place for real-time decision-making.
6. **Monitoring and Feedback**:
   - Monitor the outcomes of the implemented decisions to see if the desired results are achieved.
   - Use feedback to refine the model, improve accuracy, and adjust for changing conditions.

📊 Business Use Case Example: Airline Scheduling Optimization

Objective: Maximize revenue for an airline by optimizing flight schedules, taking into account demand, fuel costs, crew availability, and airport slots.

1. **Problem Definition**:
   - Objective: Maximize revenue while minimizing operational costs.
   - Decision Variables: Number of flights, number of passengers per flight, crew assignments, flight times.
   - Constraints: Airport availability, crew regulations (work hours), fuel costs, demand forecasts.
2. **Data Collection**:
   - Data on historical flight demand, fuel prices, crew schedules, airport congestion, competitor pricing.

3. **Model Selection**:
    - Use **Integer Programming** to decide on the best number of flights, optimal routes, and crew assignments to maximize profits.
4. **Modeling and Solution**:
    - Build an optimization model that maximizes revenue subject to constraints (e.g., number of crew members, available airport slots).
    - The model calculates the most profitable flight schedule by considering factors such as peak demand periods and low-demand seasons.
5. **Implementation**:
    - Implement the optimized schedule in the airline's booking and crew management systems.
    - Adjust flight routes, increase frequency during peak demand periods, and reduce unnecessary flights during low demand.
6. **Monitoring and Feedback**:
    - Monitor the actual revenue and operational costs post-implementation.
    - Collect feedback on customer satisfaction, flight delays, and cost savings to refine the model.

🛠 Tools and Technologies Used in Prescriptive Modeling

1. **Optimization**:
    - Gurobi, IBM ILOG CPLEX, Google OR-Tools
    - Python libraries like PuLP, SciPy, and Pyomo
2. **Simulation**:
    - @Risk (Excel), Simul8, AnyLogic
    - Python libraries like SimPy, Monte Carlo methods
3. **Decision Rules & Heuristics**:
    - Drools (open-source business rule management system)
    - Python (if-then statements, decision trees)
4. **Machine Learning**:
    - TensorFlow, Scikit-learn, PyTorch (for building AI-driven recommendation systems)

🚀 Benefits of Prescriptive Modeling

1. **Actionable Insights**: Provides clear, actionable steps for decision-makers to follow.
2. **Improved Efficiency**: Optimizes business processes, reducing waste and inefficiencies.
3. **Cost Reduction**: Minimizes operational costs while maximizing resource utilization.
4. **Better Decision-Making**: Supports complex decision-making with data-driven insights.
5. **Competitive Advantage**: Helps businesses stay ahead by making proactive and optimal decisions.

✅ 3. Linear and Non-Linear Optimization

Optimization is the process of **maximizing or minimizing an objective function** (like profit or cost) subject to **certain constraints**.

🔷 1. Linear Programming (LP)

📌 Definition:

Linear Programming (LP) involves optimizing a **linear objective function** subject to **linear constraints** (equalities or inequalities).

🧩 General Form of an LP Problem:

- Objective Function:
  Maximize or Minimize

$$Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

- Subject to Constraints:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2$$

$$x_1, x_2, \ldots, x_n \geq 0$$

🔍 Real-Life Example: Production Planning

Problem: A company produces 2 products (A and B).

- Profit from A = ₹30/unit
- Profit from B = ₹50/unit
- Each A takes 2 hours of labor and 3 kg of material
- Each B takes 4 hours of labor and 2 kg of material
- Total available: 100 hours of labor, 90 kg of material

✍️ Formulate LP:

Let

$$x_1 = \text{units of Product A}$$

$$x_2 = \text{units of Product B}$$

Objective Function:

Maximize

$$Z = 30x_1 + 50x_2$$

Subject to Constraints:

- Labor:

$$2x_1 + 4x_2 \leq 100$$

- Material:

$$3x_1 + 2x_2 \leq 90$$

- Non-negativity:

$$x_1, x_2 \geq 0$$

📈 2. Graphical Method (only for 2-variable LP)

Steps:

1. Convert inequalities to equalities.
2. Plot each line on a graph.
3. Identify the feasible region (common area satisfying all constraints).
4. Evaluate objective function ZZZ at each **corner point** of the region.
5. Choose the point with the **maximum or minimum Z value**.

Used when LP has **only two decision variables** (x1 and x2).

🔁 3. Simplex Method (for >2 variables)

Key Idea:
Iteratively moves from one vertex (solution) to a better one until the optimal solution is reached.

Steps:

1. Convert constraints to equations using slack variables.
2. Form the initial **simplex tableau**.
3. Identify **pivot column** (most negative in Z row).
4. Identify **pivot row** (minimum positive ratio).
5. Perform row operations to update the tableau.
6. Repeat until no negative values in objective row (optimal reached).

Simplex can handle **complex LP problems with multiple variables and constraints**.

Non Linear Programming

📌 Definition:

Non-Linear Programming (NLP) is a mathematical optimization technique where **either the objective function or at least one constraint is non-linear**.

💡 Common Applications of NLP:

Field	Example
Finance	Portfolio optimization with non-linear risk-return trade-offs
Engineering	Design optimization (e.g., minimizing weight of a bridge component)
Machine Learning	Cost function minimization (e.g., logistic regression)
Economics	Utility maximization, profit maximization with diminishing returns
Energy	Optimal load dispatch in power systems (non-linear demand/cost curves)

General Form of NLP:

$$\text{Minimize (or Maximize) } f(x_1, x_2, ..., x_n)$$

Subject to:

$$g_i(x_1, x_2, ..., x_n) \leq b_i \quad \text{(non-linear or linear constraints)}$$

$$x_j \geq 0$$

Example Problem:

Let's say a company wants to minimize cost of transporting goods, where the cost increases exponentially with quantity due to shipping capacity.

$$\text{Minimize } Z = x^2 + y^2 + 3xy$$
$$\text{Subject to: } x + y \leq 100, \quad x, y \geq 0$$

This is a **non-linear objective function**, so LP methods won't work.

Solution Approaches:

1. **Gradient Descent**
   - Iteratively moves toward minimum by following the slope of the function.
   - Common in machine learning.
2. **Lagrange Multipliers**
   - Used for equality constraints.
   - Solves for optimum by balancing gradients of objective and constraint functions.
3. **Karush-Kuhn-Tucker (KKT) Conditions**
   - Generalization of Lagrange for inequality constraints.
4. **Sequential Quadratic Programming (SQP)**
   - Solves a series of quadratic approximations to reach solution.
5. **Heuristics/Metaheuristics** (for complex, large-scale problems)
   - Genetic Algorithms
   - Simulated Annealing

- Particle Swarm Optimization

✅ 1. GRG (Generalized Reduced Gradient) Method

🔹 What is it?

The **GRG method** is used to solve **non-linear constrained optimization** problems. It extends the **reduced gradient method** to handle constraints efficiently.

🔧 How It Works:

- It divides the variables into:
  - **Basic (dependent) variables**
  - **Non-basic (independent) variables**
- Then it **reduces the problem** into a simpler form by eliminating the constraints.
- Uses gradient-based search to find the **optimal solution** iteratively.

✅ Applications:

- Used in Excel Solver (GRG Nonlinear engine)
- Optimization in **finance**, **marketing**, **logistics**, etc.

📌 Example Use Case:

Marketing Budget Optimization
You want to distribute your budget across different channels (TV, Online, Print) to maximize conversions. But your cost function is non-linear due to diminishing returns. GRG can solve this by iteratively adjusting allocations while satisfying budget constraints.

✅ 2. Newton's Method for Non-linear Optimization

🔹 What is it?

Newton's Method is used to find **local minima or maxima** of a non-linear function by using derivatives.

🔧 How It Works:

- Uses **first derivative (gradient)** and **second derivative (Hessian matrix)** to make updates:

$$x_{\text{new}} = x_{\text{old}} - H^{-1} \cdot \nabla f(x)$$

- Converges very quickly near the optimum (quadratic convergence), but requires computation of the **Hessian matrix**.

✅ Applications:

- Unconstrained optimization
- Used in machine learning (e.g., logistic regression parameter estimation)
- Engineering design problems

📌 Example Use Case:

ML Model Training
In logistic regression, Newton's method is used to optimize the log-likelihood function for better accuracy and faster convergence compared to gradient descent.

🧩 Convex vs Non-Convex Problems

🧩 Convex vs Non-Convex Problems

✅ 1. Convex Problems

- The **objective function and feasible region** are convex (bowl-shaped).
- **Any local minimum = global minimum.**
- Easier to solve reliably with optimization algorithms.

Example:

$$Z = x^2 + y^2 \text{ (convex function)}$$

⚠️ 2. Non-Convex Problems

- Function or constraints are not convex.
- **Multiple local minima** may exist.
- Hard to ensure the **global minimum** is found.

Example:

$$Z = x^4 - x^2 \text{ (W-shaped function)}$$

🔍 Real-Time Application: NLP in Supply Chain

A logistics company wants to **minimize delivery time** while accounting for **traffic delays (non-linear)** and **fuel consumption (non-linear relationship to load)**.

- Objective: Minimize time = f(load, distance, traffic)
- Constraints: Vehicle capacity, delivery windows
- Method: Non-linear optimization using KKT or SQP

CUTTING PLANE ALGORITHM

📌 Concept and Purpose

The **Cutting Plane Algorithm** is a mathematical method used to solve **Integer Programming (IP)** problems, especially **Mixed Integer Linear Programming (MILP)**.

- In **Integer Programming**, some or all variables must take **whole number (integer)** values.
- A **cutting plane** is a **linear constraint (cut)** added to the LP relaxation of the problem that **excludes non-integer solutions** without eliminating any feasible integer solutions.

🧠 **Goal**: Iteratively add constraints to the LP solution until an integer-only optimal solution is reached.

🧱 Why Is It Needed?

When you solve an **Integer Programming** problem using normal **Linear Programming (LP)**, the solution might include decimal values (like x = 2.7, y = 4.3), which are **invalid** for integer constraints.

The **cutting plane method** helps to:

- Eliminate fractional solutions
- Get closer to the **true integer optimal solution**

🔁 Use in Solving Integer Programming Problems

- Start by solving the **Linear Programming relaxation** (ignore integer constraints).
- If the solution is integer – great! You're done.
- If not:
    - Generate a **cutting plane** (a constraint that excludes current non-integer solution).
    - Add this new constraint to the LP and **resolve**.
    - Repeat until an integer solution is obtained.

💡 **Used with**:

- **Gomory's cutting planes** (commonly used method)
- **Branch and Cut** algorithms (modern hybrid of cutting planes and branch & bound)

⚙️ Working Mechanism – Basic Understanding

Let's break it down:

1. **LP Relaxation**:
    - Solve the original integer problem by **ignoring the integer condition**.
    - Get a solution, say x = 3.6, y = 2.1.
2. **Detect Fractional Parts**:
    - Since values aren't integers, we know this solution isn't valid.
3. **Generate a Cut**:
    - Create a new constraint (plane) that **cuts off** the current non-integer solution but **still includes all valid integer solutions**.
4. **Add Cut to the Problem**:
    - Solve the new LP with the added constraint.
5. **Repeat**:
    - Keep adding cuts until you reach a solution where all variables are integers.

🖍️ Simple Analogy

Imagine you're trying to trap a bird (integer solution) inside a cage (feasible region).

- At first, your cage has wide holes (LP relaxation) – the bird escapes.
- You slowly add **smaller wires (cuts)** to the cage so the bird can't escape, but **you don't harm it or trap something else** (no valid integer solutions are excluded).
- Eventually, the cage is tight enough that the bird (integer solution) is trapped exactly.

✅ Advantages

Advantage	Description
✅ **Systematic**	Offers a clear, step-by-step way to move toward an integer solution.
✅ **Effective for Small to Medium Problems**	Especially useful when the number of integer variables is limited.

✅ **Can Be Combined**	Works well with other techniques like branch and bound (branch-and-cut).

⚠️ Limitations

Limitation	Description
❌ **Slow for Large Problems**	Can become computationally expensive if many cuts are needed.
❌ **Difficult Cut Generation**	Creating useful cuts can be mathematically challenging.
❌ **Numerical Issues**	Repeated cuts may lead to rounding or precision errors.
❌ **Not Ideal for Non-Linear IP**	Works best with **linear** integer programming.

🧩 Real-Time Example: Warehouse Allocation

Problem: Allocate goods to 3 warehouses. You must send **whole units only**, and you want to **minimize cost**.

- LP relaxation might suggest:
  - Warehouse A: 2.5 units
  - Warehouse B: 3.5 units
- You can't send half a unit!

🔧 **Cutting Plane**:

- Add constraints to eliminate the 2.5 and 3.5 while still allowing 2 or 3.
- Eventually, reach an optimal **integer solution** like Warehouse A: 3, Warehouse B: 3.

DECISION ANALYSIS

Decision Analysis is a **structured approach to decision-making** under uncertainty. It combines data, probabilities, and logical models to help organizations or individuals choose the best course of action.

📊 1. Decision Trees

🔹 What is a Decision Tree?

A **Decision Tree** is a graphical representation of decisions and their possible consequences, including:

- **Decision nodes** (square): where a choice must be made.
- **Chance nodes** (circle): where an uncertain event happens.
- **Branches**: outcomes or choices.
- **Payoffs**: the reward/cost for each outcome.

🔧 Steps to Construct a Decision Tree:

1. Start from a **decision node**.
2. Add branches for each possible **decision**.
3. For each decision, add a **chance node** representing uncertainty.
4. From each chance node, add branches for **possible outcomes** with associated probabilities.
5. Write down the **payoffs** (profit, cost, etc.) for each final branch.
6. Use **expected values** to choose the best decision path.

✅ Real-Time Example: Launching a Product

Scenario: A company is deciding whether to **launch a new product**.

- **Decision Node**: Launch or Not Launch
- **Chance Node**: If launched → Market Response can be **High**, **Medium**, or **Low**
- **Probabilities**: High (40%), Medium (40%), Low (20%)
- **Payoffs**:
    - High: +₹10,00,000
    - Medium: +₹5,00,000
    - Low: -₹2,00,000

- Not Launch: ₹0

✅ Calculate **Expected Value** and choose the option with the highest value.

📑 2. Payoff Tables

🔹 What is a Payoff Table?

A **payoff table** shows the outcomes (profit/loss) for various **decisions** under different **states of nature** (scenarios). It's like a matrix for quick comparison.

Decision	High Demand	Medium Demand	Low Demand
Launch Product	₹10,00,000	₹5,00,000	-₹2,00,000
Don't Launch	₹0	₹0	₹0

You multiply each payoff with its **probability**, sum them, and pick the highest **expected value**.

📈 3. Expected Value of Perfect Information (EVPI)

🔹 What is EVPI?

EVPI tells us **how much we should be willing to pay** to know the future (state of nature) **with certainty**.

💡 Formula:

$$\text{EVPI} = \text{Expected Value with Perfect Information} - \text{Expected Value without Perfect Information}$$

- **Expected Value without PI (EVw/o PI)**: Max expected value using current data
- **Expected Value with PI (EVw PI)**: Assume you **know** what the future holds and always choose the best decision accordingly.

✅ Real-Time Application:

In our product launch example:

- If you knew for sure demand would be **High**, you'd definitely launch and earn ₹10,00,000.
- EVPI helps you decide whether it's worth spending money on **market research** or **AI forecasting tools**.

📉 4. Sensitivity Analysis

🔹 What is Sensitivity Analysis?

It checks **how sensitive your decision or outcome is to changes in input values** like:

- Probabilities
- Costs
- Profits

🔍 Why It Matters?

Because real-world values **are uncertain**, it's important to see:

- Will my decision still be valid if demand drops?
- What if cost increases by 20%?

✅ Real-Time Example:

In our product launch:

- If probability of **high demand drops from 40% to 20%**, is launching still the best choice?
- If cost of launch increases, does it affect the expected value?

🧩 Real-Time Business Applications of Decision Analysis

Domain	Use Case

Marketing	Decide whether to launch a campaign; use trees for ad response rates.
Healthcare	Choose between treatment plans based on patient response probabilities.
Manufacturing	Invest in new machinery vs outsourcing; payoff vs maintenance risk.
Finance	Investment decisions under market uncertainty.
Logistics	Choose best transport method under delay/weather risk.

RISK AND UNCERTAINTY METHODS

📌 1. Differences Between Risk and Uncertainty

Aspect	Risk	Uncertainty
Definition	When outcomes are unknown but **probabilities are known**	When **neither outcomes nor probabilities are known**
Data Availability	Based on historical data or statistical models	Often lacks sufficient data
Example	Tossing a coin (50/50)	Launching a never-before-seen product
Decision Approach	Use probabilistic models	Use judgment, sensitivity analysis, or scenario planning

✅ Real-Time Example:

- **Risk**: An investor knows a stock has a 70% chance of rising.
- **Uncertainty**: An entrepreneur has no idea how the market will respond to a new tech gadget.

📊 2. Probabilistic Decision Models

These models incorporate **probability distributions** to reflect real-world uncertainty.

🔧 Components:

- **Random variables**: e.g., demand, cost, delivery time
- **Probability distributions**: Normal, Poisson, Binomial, etc.
- **Expected value**: Weighted average of all outcomes

✅ **Application:**

A company deciding inventory levels can use a **probabilistic model** to simulate **customer demand** and minimize stockouts or overstocking.

🎲 **3. Monte Carlo Simulation Basics**

🔹 **What is Monte Carlo Simulation?**

Monte Carlo simulation uses **random sampling** to simulate a wide range of possible outcomes in a process that involves uncertainty.

🔁 **Steps:**

1. **Define the problem** and uncertain inputs.
2. **Assign probability distributions** to inputs (e.g., normal, uniform).
3. **Generate random samples**.
4. **Run thousands of simulations** using those samples.
5. **Analyze outcomes**: mean, standard deviation, probability of loss, etc.

✅ **Real-Time Application:**

Project Management – estimating total project cost or duration:

- Task durations are uncertain.
- Use Monte Carlo to simulate project timelines and determine the likelihood of completing before a deadline.

📈 **Tools Used:**

- Python, R, Excel with @Risk add-on, MATLAB, or Oracle Crystal Ball

---

🧮 **4. Scenario and What-If Analysis**

🔹 **Scenario Analysis:**

- Evaluates the impact of **different sets of assumptions** (scenarios) like:
    - Best case

- Worst case
- Most likely case

🔹 **What-If Analysis:**

- Tests how **changing one input** affects the outcome, holding others constant.

✅ **Real-Time Application:**

Sales Forecasting:

- Scenario Analysis:
  - Best Case: ₹1 Cr revenue (if market grows 20%)
  - Worst Case: ₹40 Lakh (if competitor launches similar product)
- What-If:
  - What if **price per unit** increases by ₹5?
  - What if **raw material cost** decreases by 10%?

Both tools help **prepare for surprises**, **stress-test decisions**, and **build flexibility** into strategy.

🧩 **Summary Table**

Method	**Purpose**	**Application Area**
Risk Analysis	Quantify impact when probabilities are known	Investment, insurance
Uncertainty Analysis	Make choices without known probabilities	New product launch, innovation
Probabilistic Models	Use probability to guide decisions	Demand planning, cost forecasting
Monte Carlo Simulation	Simulate a range of possible outcomes	Project management, finance
Scenario Analysis	Evaluate multiple hypothetical futures	Strategic planning, budgeting
What-If Analysis	Test the effect of changing a variable	Pricing, marketing, operations

---

TEXT ANALYTICS

📌 Overview of Text Mining

Text Mining (or Text Analytics) is the process of extracting meaningful information from unstructured text data. It involves:

- **Preprocessing text** (cleaning)
- **Identifying patterns and structures**
- **Generating insights or making predictions**

Since **80-90%** of business data is unstructured (e.g., emails, reviews, social media), text mining enables organizations to turn that data into actionable intelligence.

🧠 Natural Language Processing (NLP) Basics

NLP is a subfield of AI that enables machines to **understand, interpret, and respond to human language**.

🔧 Key NLP Tasks:

- **Tokenization** – splitting text into words/sentences
- **Stop word removal** – filtering common words (e.g., "is", "the")
- **Stemming/Lemmatization** – reducing words to root form
- **Named Entity Recognition (NER)** – detecting names, places, etc.
- **Part-of-Speech (POS) Tagging** – identifying nouns, verbs, adjectives

✅ Example:

Input: "The product was amazing and delivery was fast!"

- Tokens: [“product”, “amazing”, “delivery”, “fast”]
- POS Tags: noun, adjective, noun, adjective

😊 Sentiment Analysis

Sentiment Analysis detects **emotions or opinions** in text. It classifies text as:

- **Positive**
- **Negative**

- **Neutral**

🔧 **How It Works:**

- Based on **rule-based models** (using sentiment lexicons like SentiWordNet)
- Or **ML models** trained on labeled datasets

✅ **Real-Time Example:**

A customer review:

"I love the new smartphone! Battery lasts long, and it's very sleek."

→ **Sentiment**: Positive
→ **Application**: Used by brands to analyze product reception

---

💼 **Applications in Business**

Business Function	**Application**
Marketing	Analyzing customer reviews, brand sentiment, campaign feedback
Customer Support	Classifying and prioritizing support tickets based on tone and urgency
Human Resources	Analyzing employee feedback, survey results
Finance	Analyzing news for stock sentiment or fraud detection
Healthcare	Extracting symptoms from patient notes or reviews

✅ **Real-Time Business Example: Customer Reviews on Amazon**

Scenario: A smartphone company launches a new model. Thousands of customers leave reviews on Amazon.

🔹 **Step-by-Step Text Analytics:**

1. **Collect reviews** using APIs or scraping.
2. **Preprocess data** using NLP techniques.
3. **Run sentiment analysis** to classify reviews.

4. **Visualize results** (e.g., 65% Positive, 20% Neutral, 15% Negative).
5. **Take action**: Improve camera quality if many negatives are about that.

🔹 **Tools Used:**

- Python (NLTK, spaCy, TextBlob)
- Power BI or Tableau (for visualization)
- Google Cloud NLP, AWS Comprehend (for large-scale solutions)

📊 **Visualization Example:**

- Word Cloud: Most frequent words in positive/negative reviews.
- Sentiment Trend Graph: Change in sentiment over time.

WEB ANALYTICS

📌 1. Introduction to Web Analytics Tools

Web Analytics is the measurement, collection, analysis, and reporting of web data to understand and optimize web usage.

🔧 **Common Tools:**

- **Google Analytics** (most popular)
- Adobe Analytics
- Matomo (open-source)
- Hotjar (for heatmaps, session recording)
- SEMrush (for SEO/web traffic analysis)

✅ **Google Analytics (GA):**

GA tracks user behavior on websites and apps—like who visits, what they do, and how long they stay.

📊 2. Key Metrics Explained

Metric	Definition	Use Case

Page Views	Total number of times a page is viewed	Understand content popularity
Sessions	Group of interactions by a user in a single visit	Measure engagement
Users	Unique visitors to the site	Count real people, not visits
Bounce Rate	% of visitors who leave after viewing only one page	Indicates user dissatisfaction or irrelevance
Average Session Duration	How long users stay per visit	Indicates engagement level
Conversion Rate	% of users who complete a desired goal (e.g., purchase, signup)	Measures success of CTAs or campaigns
Exit Rate	% of people who leave from a particular page	Helps identify content gaps
Click-through Rate (CTR)	% of people who clicked after seeing a link or ad	Measures marketing effectiveness

---

🧭 3. Customer Journey and Clickstream Analysis

Customer Journey = Every step a user takes on the site, from entry to conversion or exit.

Clickstream Analysis = Tracking and analyzing every page clicked by users.

🔁 Process:

1. **Entry Point**: Where did they come from? (Google search, ad, email)
2. **Behavior Flow**: What path did they follow? (Home → Products → Cart)
3. **Conversion or Drop-off**: Did they purchase or abandon the process?

✅ Real-Time Application: E-commerce Site

Problem: High cart abandonment rate

Clickstream Insight:

- Many users exit at "Shipping Details" page.
- Fix: Simplify the form or provide shipping estimate earlier.

📈 4. Dashboard Creation and Data Visualization

Dashboards help **track, monitor, and present** web analytics data in a user-friendly format.

🔧 Tools:

- **Google Data Studio** (free, integrates with Google Analytics)
- Tableau / Power BI
- Looker Studio

✅ Key Dashboard Components:

- **Real-time users**
- **Traffic sources (organic, paid, direct, social)**
- **Top landing pages**
- **Conversions and funnel drop-off points**
- **Device breakdown (mobile vs desktop)**

✅ Real-Time Business Application: Online Fashion Store

🔍 Goal: Improve conversion rate from product page

1. **GA Data Shows**:
    - High bounce rate on product pages from mobile users.
    - Most exits after clicking "Size Chart."
2. **Actions Taken**:
    - Optimized mobile size chart pop-up.
    - Added more reviews and “buy now” button at the top.
3. **Result**:
    - Bounce rate dropped by 18%.
    - Conversion rate improved by 10%.

🧩 Summary

Element	Real-Time Use Case

Page Views	Identifying popular content
Bounce Rate	Diagnosing UX issues
Clickstream Analysis	Understanding user behavior flow
Conversion Rate	Measuring marketing success
Dashboard	Presenting KPIs to stakeholders
Customer Journey	Enhancing UX and reducing drop-offs