    - Autonomous vehicles navigating with imperfect sensors and unpredictable traffic.
- **Key Point**: Requires handling both incomplete information and randomness, making it computationally intensive.

Unknown state space problems:

- The agents lacks prior knowledge regarding the state spaces and transitions
- Requires online search where the agent learns the environment through interaction
- Example: In the Romania Problem, if the map is unknown, the agent must explore to discover cities and connections (e.g., learning that Arad connects to Sibiu).

Search Approach

    - Algorithms like **LRT** update knowledge during exploration.
    - Trial-and-error or reinforcement learning can map the state space over time.

- **Practical Example**:
    - A robot exploring an unmapped building, learning walls and paths as it moves.
- **Key Point**: Most complex, as the agent must simultaneously learn the environment and solve the problem.

Supervised Machine learning

Intro:

- Supervised machine learning is where the model is trained on a dataset containing input variables (features or independent variables) and corresponding output variables (labels or dependent variables).

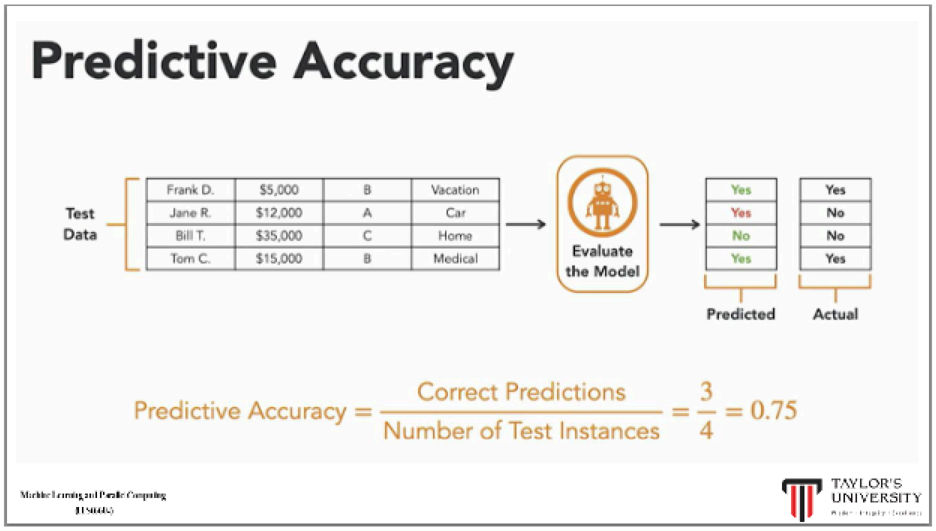
- The goal is to learn a mapping from inputs to outputs, enabling accurate predictions on new, unseen data.
- They rely on labeled data each input example is paired with a known output
- The process involves training a model to minimize prediction errors by adjusting the parameters based on training data
- two types of supervised learning are Classification (predicting categories) and Regression (predicting numerical values).

Classification

- Classification is a supervised learning task where the dependent variable is categorical meaning the model predicts discrete class labels
- The output is a class label, but some algorithms (e.g., Logistic Regression) also provide probabilities for each class.

Types of classification:

- **Binary Classification: Two classes (e.g., Default: Yes/No).**
- **Multi-class Classification: More than two classes (e.g., classifying animals as Dog, Cat, or Bird).**
-



Predictive Accuracy

Test Data

Frank D.	$5,000	B	Vacation
Jane R.	$12,000	A	Car
Bill T.	$35,000	C	Home
Tom C.	$15,000	B	Medical

Predicted	Actual
Yes	Yes
Yes	No
No	No
Yes	Yes

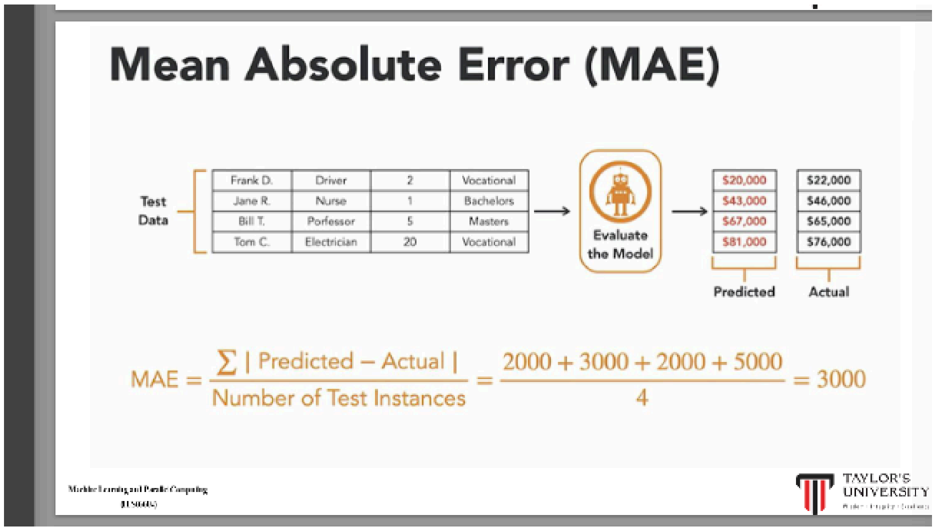
$$\text{Predictive Accuracy} = \frac{\text{Correct Predictions}}{\text{Number of Test Instances}} = \frac{3}{4} = 0.75$$

Machine Learning and Parallel Computing [illegible]

TAYLOR'S UNIVERSITY

Regression:

- Regression is a supervised learning tasks where the dependent variable is continuous meaning the model predicts numerical values
- The model learns to find patterns in the data that map input features to a continuous output.
- Types of regression
    - SLP:
        - One independent variable and one dependent variable
    - MLP
        - Multiple independent variable and one dependent
    - Polynomial Regression
        - Models non-linear relationships by using polynomial terms
    - Poisson Regression:
        - Used for count data (e.g., number of customer complaints).
- predicts continuous numerical outcomes.
- Common algorithms: Simple Linear Regression, Multiple Linear Regression, Polynomial Regression, Poisson Regression.
- Evaluation metric: Mean Absolute Error (MAE) for measuring prediction error.



Mean Absolute Error (MAE)

Test Data

Frank D.	Driver	2	Vocational
Jane R.	Nurse	1	Bachelors
Bill T.	Porfessor	5	Masters
Tom C.	Electrician	20	Vocational

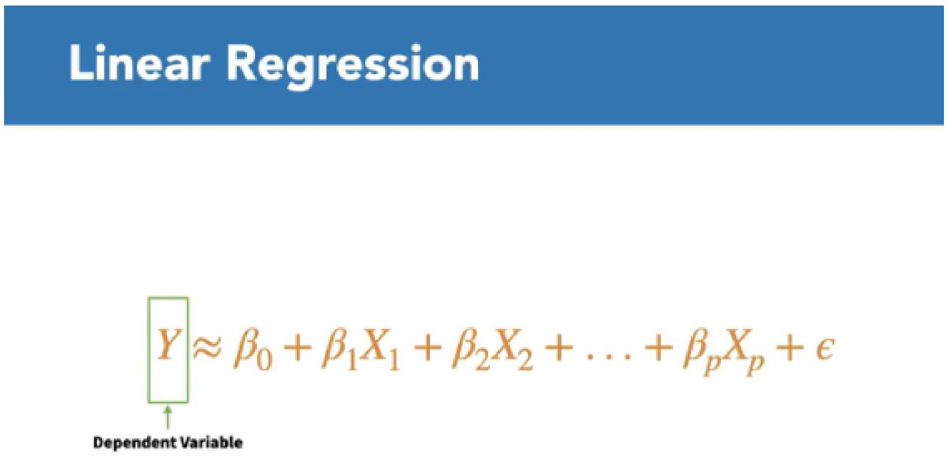
Predicted	Actual
$20,000	$22,000
$43,000	$46,000
$67,000	$65,000
$81,000	$76,000

$$\text{MAE} = \frac{\sum |\text{Predicted} - \text{Actual}|}{\text{Number of Test Instances}} = \frac{2000 + 3000 + 2000 + 5000}{4} = 3000$$

Key algorithms in Supervised Learning

Linear Regression:

- Linear Regression models the relationship between a dependent variable and one for more independent variables using a straight line.
- The model assumes a linear relationship between inputs and outputs, represented by the equation y=mx+b, where y is the dependent variable, x is the independent variable, m is the slope, and b is the intercept.
- For multiple linear regression, the equation becomes y=b0+b1×1+b2×2+…+bnxn y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n y=b0+b1×1+b2×2+…+bnxn, where each x is an indepeny variable and b is its coefficient
- Limitations: Assumes linearity, which may not hold for complex data.
- Used for regression tasks with continuous outputs.
- Simple and interpretable but may underperform on non-linear data.



Linear Regression

$$\underbrace{\boxed{Y}}_{\text{Dependent Variable}} \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p + \epsilon$$

Logistic Regression:

- Logistic regression models the relationship between independent variables and a categorical dependent variable using a sigmoid curve
- Unlike linear regression, logistic regression predicts probabilities for categorical outcomes.

- The log-odds formula (Page 18): log(p1−p)=b0+b1×1+b2×2+... \log\left(\frac{p}{1-p}\right) = b_0 + b_1x_1 + b_2x_2 + \dots log(1−pp)=b0+b1×1+b2×2+..., where p p p is the probability of the positive class.

  • Example: Predicting whether a customer defaults (Yes/No) based on loan amount and credit grade.
- Used for binary classification (can be extended to multi-class).
- Outputs probabilities, making it interpretable for decision-makin

$$\text{Log Odds} = \underbrace{\boxed{\log\left(\frac{P}{1-P}\right)}}_{\text{Logit}} = \beta_0 + \beta_1 X$$

Decision Trees:]

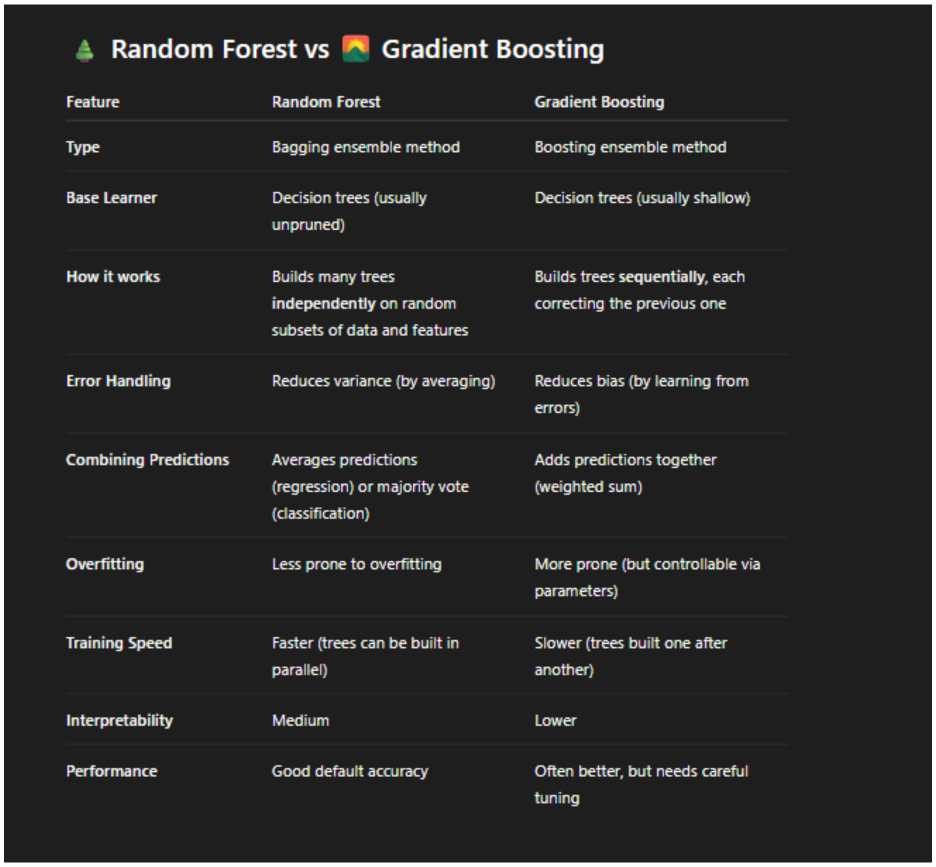
- a tree like model that makes decision by recursively splitting the input space based on feature values
- Each node represents a decision based on a feature leading to branches and eventually a leaf node with a class label or value
- Advantages: Easy to visualize and interpret.
- Disadvantages: Prone to overfitting, especially with deep trees.
- Used for both classification and regression.
- Intuitive but requires pruning to avoid overfitting

Random Forests:

- An ensemble method that combines multiple decision trees to improve predictive accuracy and reduce overfitting
- Random Forests use bagging (bootstrap aggregating) where multiple trees are trained on random subsets of the data and features
- The final prediction made by averaging or voting (regression or classification)
- advantages:
    - Robust to overfitting and handles high-dimensional data well.
    - Improves accuracy over single decision trees.
- Cons:
    - Computationally intensive but effective for complex datasets.

Gradient Boosting Machines:

- An ensemble method that builds trees sequentially with each tree correcting errors of the previous ones
- Unlike Random Forests, Gradient Boosting uses boosting, where trees are built iteratively, and each tree focuses on reducing the errors of the previous ones.
- Advantages: High accuracy, especially for structured data.
- Disadvantages: Sensitive to hyperparameters and computationally expensive.
- Sequentially improves predictions.
- Popular implementations: XGBoost, LightGBM, CatBoost.



🌲 Random Forest vs 🌄 Gradient Boosting

Feature	Random Forest	Gradient Boosting
Type	Bagging ensemble method	Boosting ensemble method
Base Learner	Decision trees (usually unpruned)	Decision trees (usually shallow)
How it works	Builds many trees **independently** on random subsets of data and features	Builds trees **sequentially**, each correcting the previous one
Error Handling	Reduces variance (by averaging)	Reduces bias (by learning from errors)
Combining Predictions	Averages predictions (regression) or majority vote (classification)	Adds predictions together (weighted sum)
Overfitting	Less prone to overfitting	More prone (but controllable via parameters)
Training Speed	Faster (trees can be built in parallel)	Slower (trees built one after another)
Interpretability	Medium	Lower
Performance	Good default accuracy	Often better, but needs careful tuning

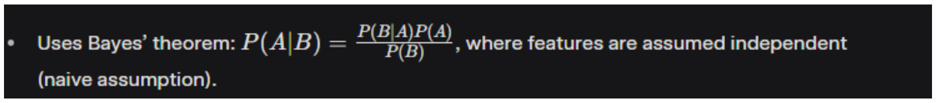
K-nearest Neighbors:

- classifies data points based on the majority class of their k nearest neighbors
- KNN is a lazy learning algorithm meaning no explicit training phase occurs ; it stores the training data and then later computes distances during prediction
- Distance metrics: Euclidean, Manhattan, etc.
- Disadvantages: Slow for large datasets and sensitive to the choice of k.
- Simple and non-parametric.
- Performance depends on distance metric and k.

Naive Bayes:

- A probabilistic classifier based on bayes theorem assuming feature independence
- Efficient for classification tasks.

- Works well with categorical or sparse data.



- Uses Bayes' theorem: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$, where features are assumed independent (naive assumption).

- Advantages: Fast and effective for text data or small datasets.
- Disadvantages: Independence assumption may not hold in real-world data

Support Vector Machines (SVM)

- finds the optimal hyperplane to separate classes with the maximum margin
- SVM aims to maximize the margin between classes using support vectors (data points closest to the hyperplane)
- For non-linear data, SVM uses the kernel trick to transform into a higher dimensional space
- Advantages: Effective in high-dimensional spaces.
- Disadvantages: Sensitive to scaling and computationally intensive for large datasets.
- Maximizes margin for robust classification.
- Kernel functions enable non-linear classification

The machine Learning Process:

- The workflow for building a supervised machine learning model from data collection to prediction
- **Data Collection**: Gather a labeled dataset with features and outcomes. Example (Page 10): Loan dataset with Customer, Amount, Grade, Purpose, and Default.
- **Data Preprocessing**: Clean data (handle missing values, normalize features), encode categorical variables (e.g., Grade: A, B, C), and split into training and test sets.

- **Model Training**: Train the chosen algorithm (e.g., Logistic Regression) on the training data to learn patterns.
- **Evaluation**: Assess the model on test data using metrics like Predictive Accuracy (classification) or MAE (regression).
- **Prediction**: Use the trained model to predict outcomes for new data.
- The process is iterative, often requiring hyperparameter tuning, feature engineering, and model selection.
    - Iterative process: Collect → Preprocess → Train → Evaluate → Predict.

Evaluation Metrics

Predictive accuracy

- Measures the proportion of correct predictions in a classification task
-
- Limitations: May be misleading for imbalanced datasets (e.g., if 95% of loans are non-defaults, predicting "No" always gives high accuracy).
- Alternative metrics: Precision, Recall, F1-Score (not mentioned in slides but relevant).

Formula: $$\text{Predictive Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Number of Test Instances}}.$$

MAE

- Measures the average absolute different between predicted and actual value in a regression task
- Advantages : Simple and interpretable robust to outliers compared to mean squared error (MSE)

- Formula: $$\text{MAE} = \frac{\sum |\textit{Predicted} - \text{Actual}|}{\text{Number of Test Instances}}.$$

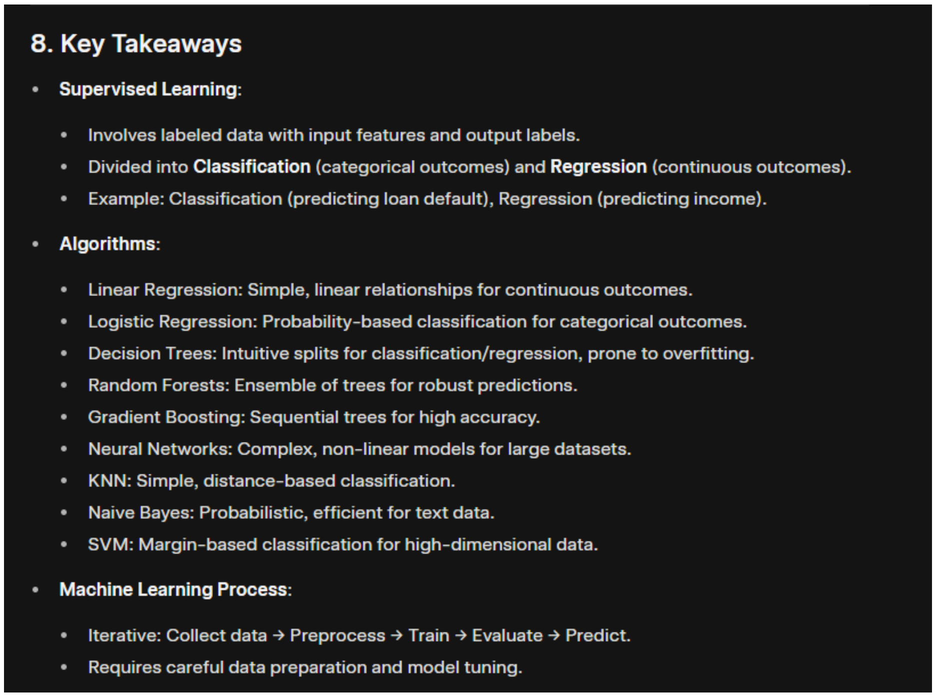
Unsupervised Learning:

K-means clustering:

- Groups data into k clusters on similarity where k is predefined
- Algorithm iteratively assign data points to clusters by minimizing the distance to cluster
- Unsupervised method for unlabelled data.
- Requires choosing k and distance metric (e.g., Euclidean).

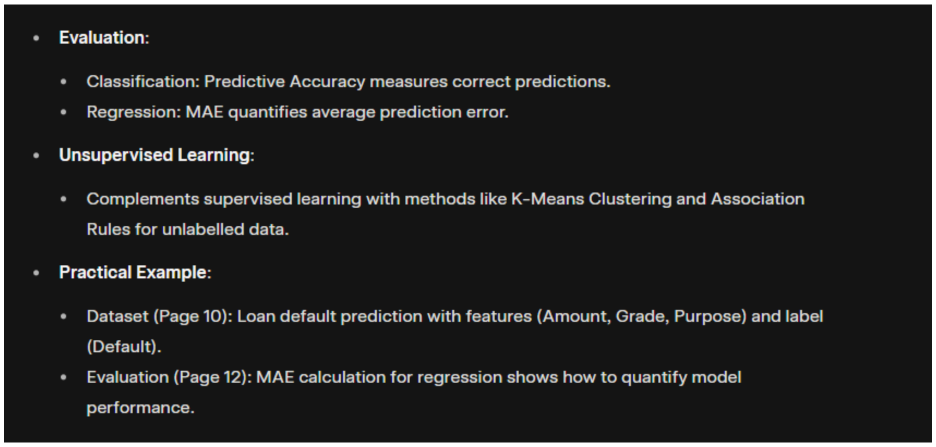
Association rules:

- Identifies pattern of co-occurrence in data using if then statements
- Metrics: Support (frequency of rule), Confidence (strength of rule), Lift (rule effectiveness).
- Applications: Recommendation systems, cross-selling strategies.



8. Key Takeaways

- **Supervised Learning**:
    - Involves labeled data with input features and output labels.
    - Divided into **Classification** (categorical outcomes) and **Regression** (continuous outcomes).
    - Example: Classification (predicting loan default), Regression (predicting income).
- **Algorithms**:
    - Linear Regression: Simple, linear relationships for continuous outcomes.
    - Logistic Regression: Probability-based classification for categorical outcomes.
    - Decision Trees: Intuitive splits for classification/regression, prone to overfitting.
    - Random Forests: Ensemble of trees for robust predictions.
    - Gradient Boosting: Sequential trees for high accuracy.
    - Neural Networks: Complex, non-linear models for large datasets.
    - KNN: Simple, distance-based classification.
    - Naive Bayes: Probabilistic, efficient for text data.
    - SVM: Margin-based classification for high-dimensional data.
- **Machine Learning Process**:
    - Iterative: Collect data → Preprocess → Train → Evaluate → Predict.
    - Requires careful data preparation and model tuning.



- **Evaluation:**
  - Classification: Predictive Accuracy measures correct predictions.
  - Regression: MAE quantifies average prediction error.
- **Unsupervised Learning:**
  - Complements supervised learning with methods like K-Means Clustering and Association Rules for unlabelled data.
- **Practical Example:**
  - Dataset (Page 10): Loan default prediction with features (Amount, Grade, Purpose) and label (Default).
  - Evaluation (Page 12): MAE calculation for regression shows how to quantify model performance.

DT

- A decision tree is a predictive model that maps features to outcomes through a series of decision rules. It splits data into subsets based on feature thresholds, creating a tree where:
  -
  - **Root Node**: The starting point, representing the most significant feature for splitting (e.g., "Does the job pay more than $80,000 per year?" on Page 5).
  - **Internal Nodes**: Represent subsequent decision points based on other features (e.g., "Is the commute less than an hour?" on Page 6).
  - **Branches**: Represent the outcome of a decision (e.g., Yes/No paths).
  - **Leaf Nodes**: Represent the final prediction (e.g., "Accept" or "Reject" on Page 5).
- **Transparency**: Decision trees are "white-box" models, meaning their logic is easily interpretable, unlike "black-box" models like neural networks (Page 9).
- **Key Point**: Decision trees are intuitive, making them suitable for applications requiring clear decision-making processes, such as business or policy decisions.

Types of decision trees:

Classification:

- Definition: Used when the dependent variable is categorical, such as predicting whether a customer will default on a loan (Yes/No).
- How It Works: The tree splits data based on conditions
- Key Point: Classification trees are effective for binary or multi-class problems, producing categorical outcomes through recursive splitting.

Regression

- Definition: Used when the dependent variable is continuous, such as predicting a customer's income.
- How It Works: The tree splits data to minimize variance in the continuous outcome (e.g., average income in a leaf node).
- Key Point: Regression trees predict numerical values by averaging the dependent variable within each leaf node, suitable for tasks like income or price prediction.

Recursive partitioning:

Process:

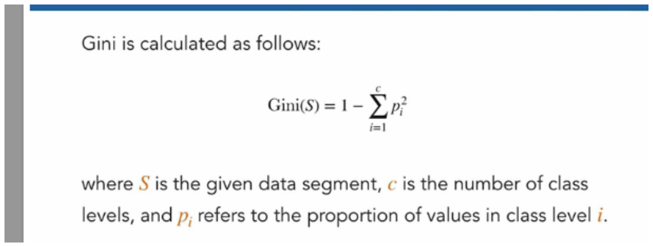
- its the process where data is repeatedly split into subsets based on feature thresholds to maximize homogeneity in each partition

Steps

1. Select a feature and threshold that best separates the data (e.g., Income > $25,000).
2. Split the data into two or more subsets.
3. Repeat for each subset until a stopping criterion is met:
    - All data in a partition belong to the same class (classification) or have similar values (regression).
    - All features are exhausted.
    - A user-defined limit (e.g., maximum tree depth) is reached (Page 31).

optimization metrics :

- splits are chosen to minimize impurity for classification or variance for regression
- Classification: Common metrics include Gini impurity or entropy.
  - Gini impurity is the measure statistical dispersion, higher the randomness higher teh gini



Gini is calculated as follows:

$$\text{Gini}(S) = 1 - \sum_{i=1}^{c} p_i^2$$

where S is the given data segment, c is the number of class levels, and p_i refers to the proportion of values in class level i.

- Regression: Sum of Squared Residuals (SSR) measures variance within partitions
  - SSR is the quantification of difference between partition and average value

$$\text{SSR} = \sum_{i=1}^{n} (y_i - \hat{y})^2$$

- Key Point: Recursive partitioning creates a tree by iteratively splitting data to maximize class purity or minimize variance, guided by metrics like Gini, entropy, or SSR.

Overfitting and pruning:]

- Overfitting (Page 32): Occurs when the tree is too complex, capturing noise in the training data rather than general patterns. This leads to poor performance on unseen data.

Pre-pruning

- Limits tree growth during construction (e.g., set maximum depth, minimum samples per leaf).

- **Advantage**: Computationally efficient.
- **Disadvantage**: May miss important patterns by stopping splits too early.

Post-pruning

- Builds a full tree, then removes branches that contribute little to predictive accuracy.
- **Advantage**: Captures more patterns initially, then simplifies.
- **Disadvantage**: More computationally intensive.

Pruning balances model complexity and generalizability, preventing overfitting while maintaining predictive power.

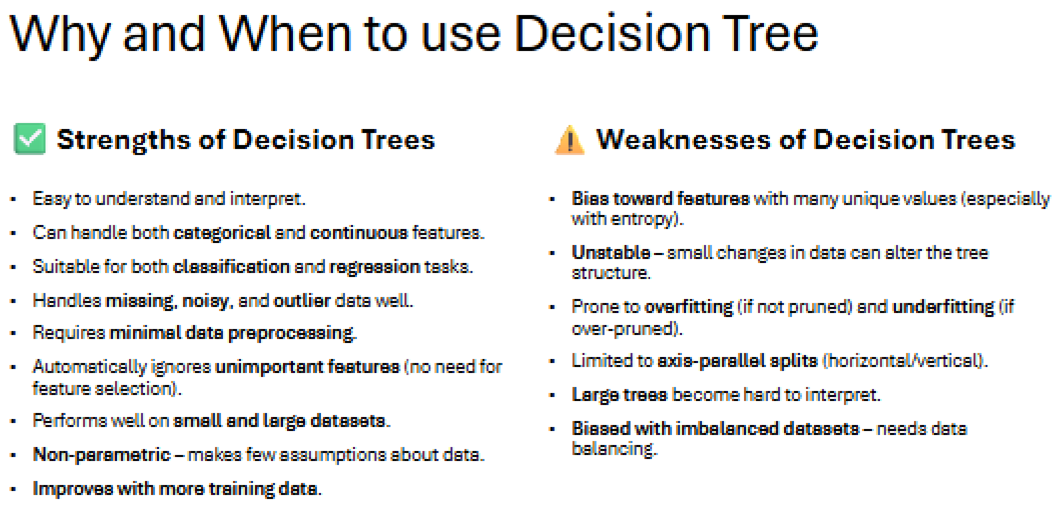
Strength and weakness:

- **Strengths**:
    - **Interpretable**: Easy to visualize and explain (e.g., job acceptance tree: Salary > $80,000 → Accept).
    - **Flexible**: Handles both categorical (e.g., Grade) and continuous (e.g., Income) features.
    - **Versatile**: Applicable to classification (e.g., loan default) and regression (e.g., income prediction).
    - **Robust**: Handles missing data, noise, and outliers well.
    - **Minimal Preprocessing**: No need for feature scaling or extensive data cleaning.
    - **Automatic Feature Selection**: Ignores irrelevant features during splitting.
    - **Scalable**: Works well with small or large datasets.
    - **Non-parametric**: Makes few assumptions about data distribution.
    - **Data-Driven**: Improves with more training data.
- **Weaknesses**:
    - **Bias Toward Features with Many Values**: Features with more unique values (e.g., continuous variables) may dominate splits, especially with

entropy.

- **Unstable**: Small changes in data can lead to different tree structures.
- **Overfitting/Underfitting**: Overfitting occurs without pruning; over-pruning causes underfitting.
- **Axis-Parallel Splits**: Limited to horizontal/vertical splits, which may not capture complex patterns.
- **Interpretability Issues**: Large trees become complex and hard to interpret.
- **Imbalanced Data Bias**: Performs poorly on imbalanced datasets unless balanced (e.g., oversampling minority class).

- **Key Point**: Decision trees are powerful and interpretable but require careful tuning (e.g., pruning, balancing) to address instability and overfitting.



Why and When to use Decision Tree

✅ **Strengths of Decision Trees**

- Easy to understand and interpret.
- Can handle both **categorical** and **continuous** features.
- Suitable for both **classification** and **regression** tasks.
- Handles **missing, noisy**, and **outlier** data well.
- Requires **minimal data preprocessing**.
- Automatically ignores **unimportant features** (no need for feature selection).
- Performs well on **small and large datasets**.
- **Non-parametric** – makes few assumptions about data.
- **Improves with more training data**.

⚠️ **Weaknesses of Decision Trees**

- **Bias toward features** with many unique values (especially with entropy).
- **Unstable** – small changes in data can alter the tree structure.
- Prone to **overfitting** (if not pruned) and **underfitting** (if over-pruned).
- Limited to **axis-parallel splits** (horizontal/vertical).
- **Large trees** become hard to interpret.
- **Biased with imbalanced datasets** – needs data balancing.

2. K-Nearest Neighbor (KNN) Algorithm

KNN is a non-parametric, instance-based learning algorithm used for classification and regression tasks.

Key Points:

- **Definition**: KNN predicts the class or value of a new data point by considering the K closest data points (neighbors) based on a distance metric (Page 39).
- **Process** (Page 45):

1. Calculate distance between the new data point and all training points.
2. Identify the K closest instances (neighbors).
3. Make predictions:
    - **Classification**: Majority vote among K neighbors' labels.
    - **Regression**: Average of K neighbors' values.

- **Key Point**: KNN is simple but computationally intensive for large datasets.

Common Distance Measures (Pages 40-41):

- **Euclidean**: Straight-line distance between two vectors.
- **Hamming**: Difference between binary vectors.
- **Manhattan**: Sum of absolute differences between vectors.
- **Minkowski**: Generalization of Euclidean and Manhattan distances.
- **Key Point**: The choice of distance measure impacts KNN's performance; Euclidean is most common for continuous data.

Choosing the Value of K (Pages 42-43):

- **Importance of K**: Determines the number of neighbors considered for prediction (Page 42).
- **How to Find K** (Page 43):
    1. Run KNN multiple times with different K values.
    2. Select K that minimizes error while maintaining accurate predictions.
    3. Best practice: Use an odd K to avoid ties in majority voting (classification).
- **Key Point**: K is a hyperparameter tuned via trial and error or methods like GridSearch or Random Search (Page 46).

Advantages of KNN (Page 44):

- Easy to implement.
- Non-parametric (no assumptions about data distribution).

- Versatile (works for classification and regression).

Disadvantages of KNN (Page 44):

- Slow, especially for large datasets (computes distances for all points).
- Impractical for rapid predictions.

Performance Metrics (Page 46):

- **Accuracy**: Proportion of correct predictions.
- **Precision**: Proportion of positive predictions that are correct.
- **Recall**: Proportion of actual positives correctly identified.
- **F1 Score**: Harmonic mean of precision and recall.
- **Key Point**: These metrics evaluate KNN's performance; GridSearch or Random Search can optimize K and other parameters.

: Probability and Bayesian Networks

1. Probability

- **Definition**: Probability is a mathematical framework for quantifying uncertainty, assigning values from 0 (impossible) to 1 (certain) to events. It provides a way to reason about uncertain outcomes in AI systems.
- **Key Concepts**:
  - **Events**: Outcomes of interest (e.g., it might rain tonight, Page 4).
  - **Probability Distributions**: Describe likelihoods for all possible outcomes (e.g., 10% chance of rain).
  - **Joint Probability**: Probability of multiple events occurring together, e.g., (P(\text{rain}, \text{cloudy})).
- **Example**: The document's weather forecast (Page 4) uses probabilities to predict a 10% chance of rain, reflecting uncertainty in future conditions.
- **Applications**:

    - Risk assessment (e.g., predicting equipment failure).
    - Decision-making under uncertainty (e.g., choosing whether to carry an umbrella).

2. Bayesian Networks

- **Definition**: A Bayesian network is a graphical model that represents a set of random variables and their probabilistic dependencies using a directed acyclic graph (DAG) (Page 1). Nodes represent variables, and edges indicate conditional dependencies.
- **Components**:
    - **Nodes**: Random variables (e.g., "Rain," "Cloudy").
    - **Directed Edges**: Represent causal or conditional relationships (e.g., Cloudy → Rain means rain depends on cloudiness).
    - **Conditional Probability Distributions (CPDs)**: For each node, a CPD specifies the probability of the node's value given its parents, e.g., (P(\text{Rain} | \text{Cloudy})).
- **Purpose**: Bayesian networks compactly model joint probability distributions, enabling efficient probabilistic inference and reasoning under uncertainty.
- **How They Work**:
    - Use Bayes' theorem: (P(A|B) = \frac{P(B|A)P(A)}{P(B)}), where:
        - (P(A|B)): Posterior probability (updated belief after evidence).
        - (P(B|A)): Likelihood (probability of evidence given hypothesis).
        - (P(A)): Prior probability (initial belief).
        - (P(B)): Evidence probability (normalizing constant).
    - Example: If clouds are observed (evidence), infer the probability of rain (hypothesis).
- **Advantages**:
    - Reduces computational complexity by exploiting conditional independencies.

    - Intuitive for modeling real-world relationships (e.g., symptoms and diseases).
- **Challenges**:
    - Constructing accurate CPDs requires domain expertise or data.
    - Inference in large networks can be computationally intensive.
- **Applications**:
    - Medical diagnosis (inferring diseases from symptoms).
    - Fault detection in systems (e.g., identifying causes of sensor failures).

3. Markov Assumption

- **Definition**: The Markov assumption states that the current state of a system depends only on a finite number of previous states, typically the most recent one (Page 112).
    - **First-Order Markov**: The current state depends only on the previous state,
    - **Higher-Order Markov**: The current state depends on multiple previous states,
- **Intuition**: The system has a “short memory,” ignoring distant history to simplify modeling.
- **Purpose**: Reduces computational complexity by limiting the number of dependencies, making it feasible to model dynamic systems.
- **Limitations**:
    - May oversimplify systems with long-term dependencies (e.g., climate trends affecting weather).
    - Assumes stationarity (transition probabilities don’t change over time), which may not hold in all cases.
- **Applications**:
    - Modeling sequences like stock prices, user behavior, or game states.
    - Simplifying temporal models in AI and robotics.

4. Markov Chains

- **Definition**: A Markov chain is a sequence of random variables where the probability of each state depends only on the previous state, adhering to the Markov assumption (Page 114, "Markov cian" likely means Markov chain).
- **Components**:
  - **States**: Possible configurations of the system (e.g., sunny, cloudy, rainy).
  - **Transition Model**: A matrix or function specifying probabilities of moving from one state to another (Page 115), e.g., (P(\text{rainy}_{t+1} | \text{sunny}_t) = 0.1).
- **Properties**:
  - **Memoryless**: Future states depend only on the current state (first-order Markov).
  - Can be **discrete** (finite states) or **continuous** (infinite states).
  - **Stationary**: Transition probabilities remain constant over time (common assumption).
- **Applications**:
  - Weather prediction (Page 4).
  - Natural language processing (e.g., predicting the next word).
- **Challenges**:
  - Estimating transition probabilities requires sufficient data.
  - Limited by the Markov assumption for complex systems.

5. Hidden Markov Models (HMMs)

- **Definition**: A Hidden Markov Model is a probabilistic model where the system's states are hidden (not directly observable), but observations are probabilistically dependent on these states (Page 119, "Hidden Marlow MoOGelS" likely means HMMs).
- **Components**:

    - **Hidden States**: Unobservable states of the system (e.g., true weather conditions like sunny or rainy) (Page 118).
    - **Observations**: Visible outputs generated by hidden states (e.g., sensor readings indicating rain) (Page 118).
    - **Transition Model**: Probabilities of transitioning between hidden states (Page 115), e.g., (P(\text{rainy}_{t+1} | \text{sunny}_t)).
    - **Sensor Model**: Probabilities of observations given hidden states (Page 121), e.g., (P(\text{sensor = wet} | \text{rainy})).
    - **Initial State Distribution**: Probabilities of starting in each hidden state.
- **Sensor Model Assumption**: Observations depend only on the current hidden state, not previous states or other observations (Page 122, "sensor narrow assuming" likely means this assumption).
- **How HMMs Work**:
    - The system evolves through hidden states according to the transition model.
    - Each hidden state generates an observation according to the sensor model.
    - Inference algorithms (e.g., Viterbi, Forward-Backward) estimate hidden states or predict future observations.
- **Applications**:
    - **Speech Recognition**: Hidden states are phonemes; observations are audio signals.
    - **Bioinformatics**: Hidden states are DNA sequences; observations are protein alignments.
    - **Robotics**: Inferring location (hidden state) from sensor data (observations).
- **Challenges**:
    - Parameter estimation (transition and sensor models) is complex with limited or noisy data.

    - Scalability issues for large state spaces or long sequences.
    - Assumes Markov property, which may not capture long-term dependencies.

6. Transition Model

- **Definition**: The transition model specifies the probability of moving from one state to another in a Markov process or HMM (Page 115).
- **Role**: Captures the dynamics of how a system evolves over time, enabling prediction of future states.
- **Importance**:
    - Essential for modeling temporal evolution in Markov chains and HMMs.
    - Enables forecasting in dynamic systems like weather or stock markets.
- **Challenges**:
    - Requires accurate estimation from historical data.
    - Non-stationary systems (where probabilities change over time) are harder to model.

7. Sensor Model

- **Definition**: The sensor model defines the probability of an observation given a hidden state in an HMM (Page 121, "Sensor Mode"; Page 117, "sense madale" likely means sensor model).
- **Role**: Bridges hidden states to observable data, accounting for noise or errors in sensors or measurements.
- **Assumption**: Observations are conditionally independent of other states and observations given the current hidden state (Page 122, "sensor narrow assuming").
- **Representation**:
    - As a probability distribution, e.g., (P(\text{observation} | \text{state})).

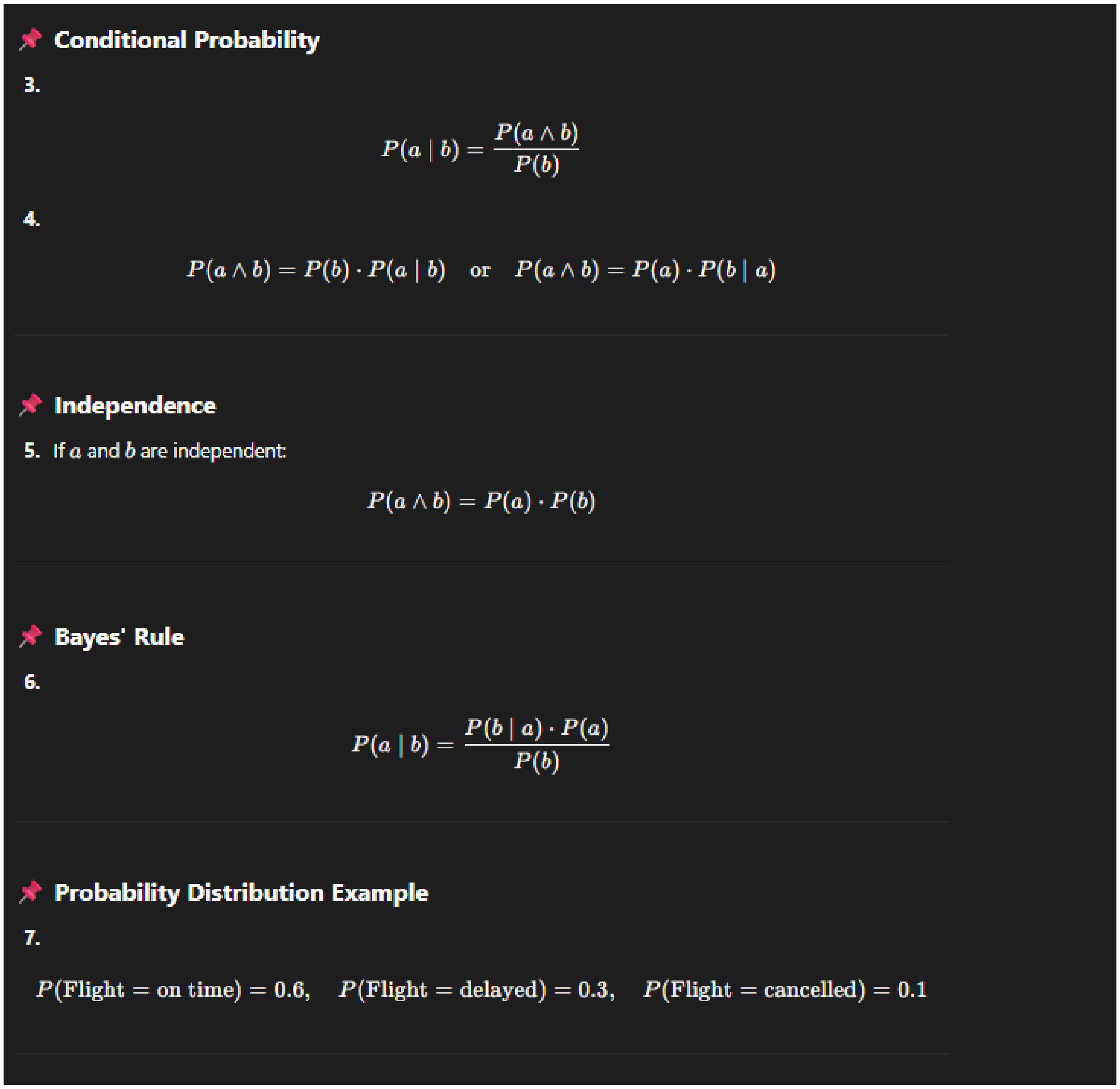
    - Example: (P(\text{sensor = wet} | \text{rainy}) = 0.9), (P(\text{sensor = wet} | \text{sunny}) = 0.1).
- **Challenges**:
    - Modeling sensor noise accurately requires calibration or data.
    - Handling multiple sensors or conflicting observations increases complexity.

8. Uncertainty Over Time

- **Definition**: Uncertainty over time refers to the challenge of predicting future states in dynamic systems where transitions and observations are probabilistic (Page 110).
- **Key Idea**:
    - Systems evolve unpredictably due to stochastic transitions (modeled by transition models).
    - Observations are noisy or incomplete (modeled by sensor models).
- **Modeling Approaches**:
    - **Markov Chains**: For fully observable systems, predict future states directly (e.g., weather transitions, Page 4).
    - **Hidden Markov Models**: For partially observable systems, infer hidden states and predict future observations (Page 119).
- **Challenges**:
    - **High-Dimensional State Spaces**: Large numbers of states increase computational complexity.
    - **Long-Term Predictions**: Uncertainty grows over time, reducing accuracy.
    - **Non-Stationarity**: Real-world systems may have changing probabilities.

12. Connections and Insights

- **Probability and Bayesian Networks**: Provide the foundation for modeling uncertainty, used in static systems (Bayesian networks) or dynamic systems (Markov models).
- **Markov Models vs. Bayesian Networks**:
  - Markov models (chains, HMMs) focus on temporal sequences, leveraging the Markov assumption.
  - Bayesian networks model static relationships with arbitrary dependencies.
  - Example: A Bayesian network might model rain depending on humidity and temperature; an HMM models rain evolving over days (Page 4).
- **HMMs as Dynamic Bayesian Networks**: HMMs can be viewed as a special case of Bayesian networks unrolled over time, with hidden states and observations linked by transition and sensor models.
- **Limitations**:
  - Markov assumption may fail for systems with long-term memory (e.g., climate affecting weather).
  - HMMs assume stationarity, which may not hold in evolving systems.
  - Scalability issues arise in high-dimensional or complex models.



📌 Conditional Probability

3.

$$P(a \mid b) = \frac{P(a \wedge b)}{P(b)}$$

4.

$$P(a \wedge b) = P(b) \cdot P(a \mid b) \quad \text{or} \quad P(a \wedge b) = P(a) \cdot P(b \mid a)$$

📌 Independence

5. If a and b are independent:

$$P(a \wedge b) = P(a) \cdot P(b)$$

📌 Bayes' Rule

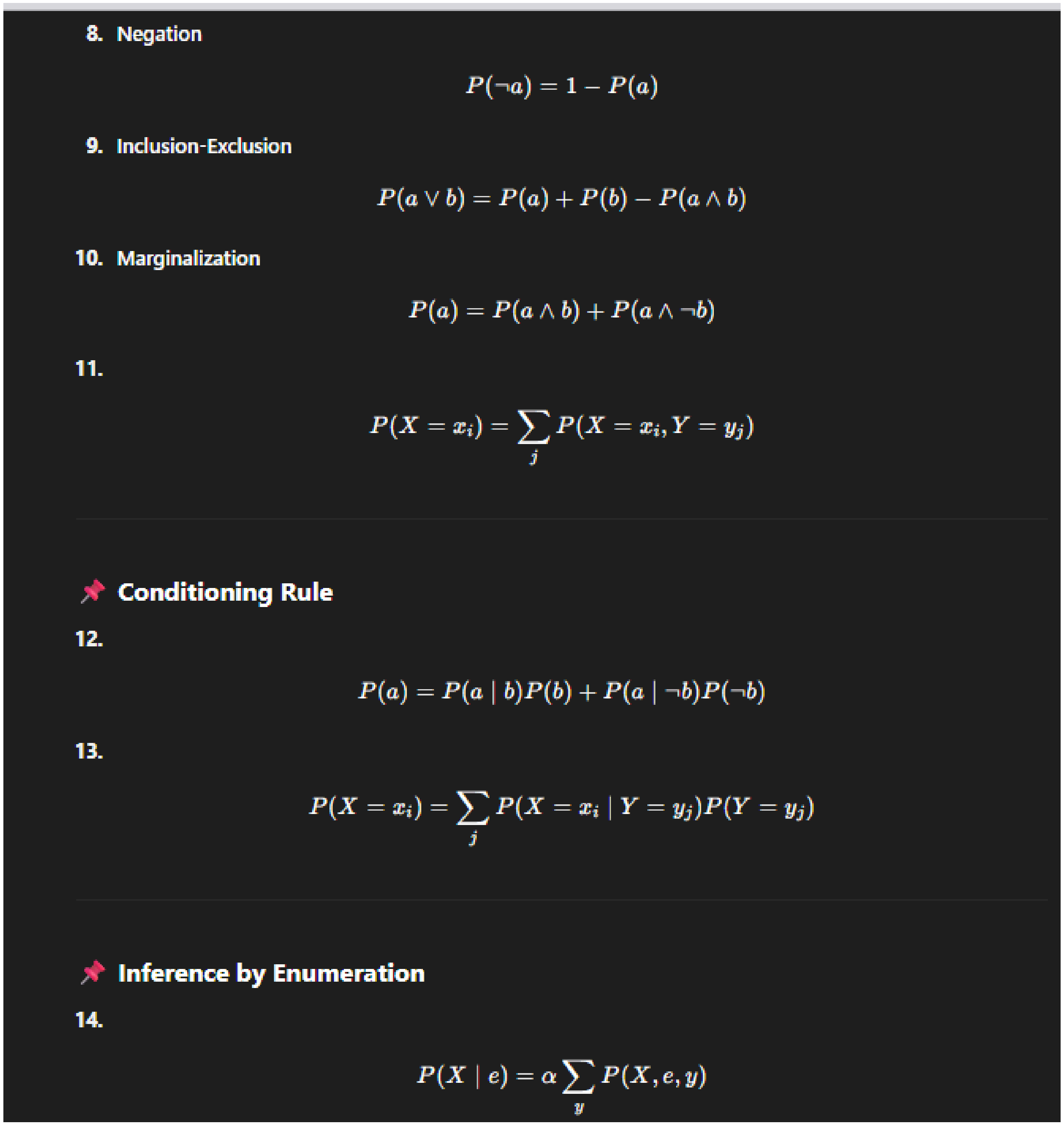
6.

$$P(a \mid b) = \frac{P(b \mid a) \cdot P(a)}{P(b)}$$

📌 Probability Distribution Example

7.

$$P(\text{Flight} = \text{on time}) = 0.6, \quad P(\text{Flight} = \text{delayed}) = 0.3, \quad P(\text{Flight} = \text{cancelled}) = 0.1$$



8. **Negation**

$$P(\neg a) = 1 - P(a)$$

9. **Inclusion-Exclusion**

$$P(a \vee b) = P(a) + P(b) - P(a \wedge b)$$

10. **Marginalization**

$$P(a) = P(a \wedge b) + P(a \wedge \neg b)$$

11.

$$P(X = x_i) = \sum_j P(X = x_i, Y = y_j)$$

📌 Conditioning Rule

12.

$$P(a) = P(a \mid b)P(b) + P(a \mid \neg b)P(\neg b)$$

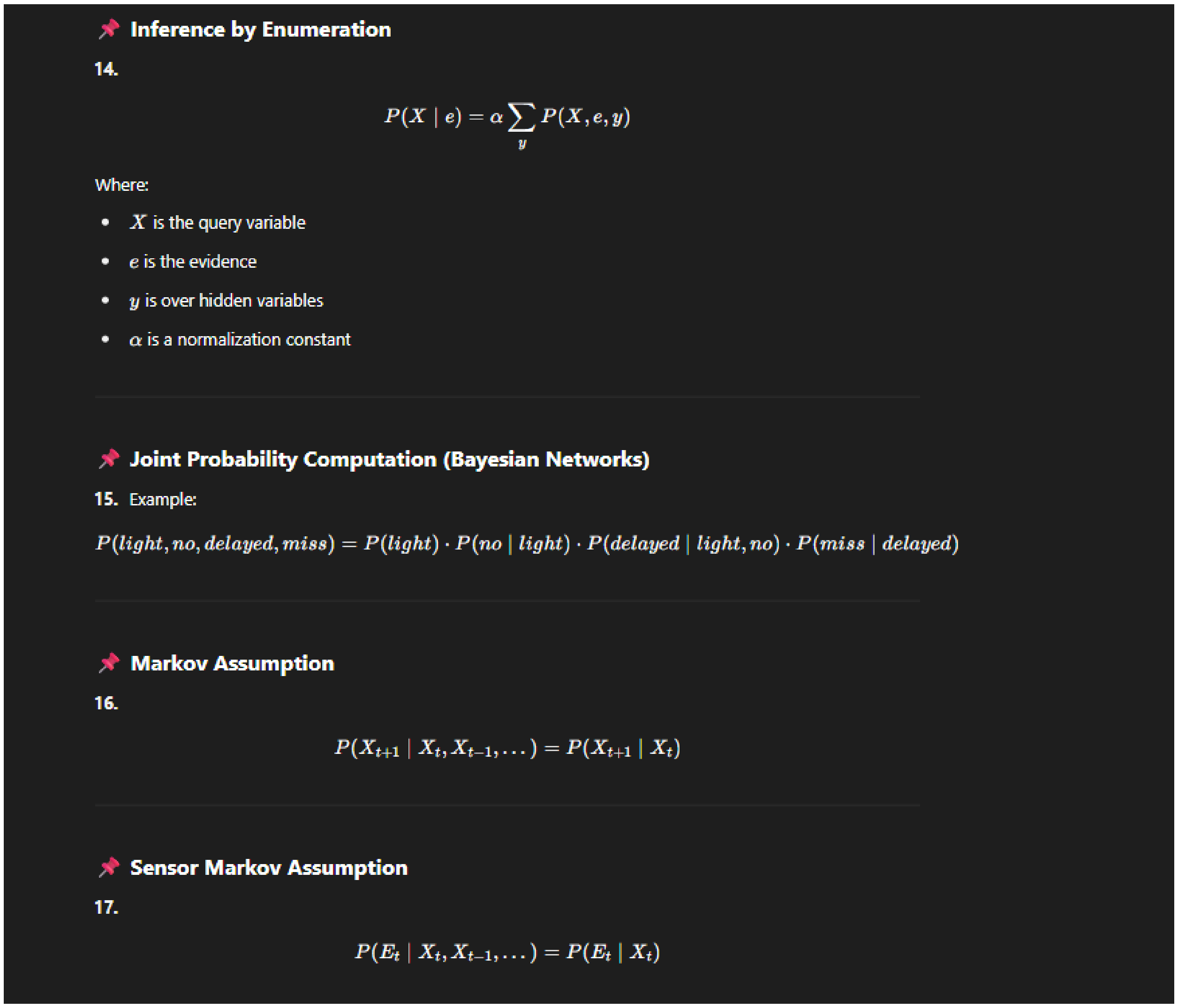
13.

$$P(X = x_i) = \sum_j P(X = x_i \mid Y = y_j)P(Y = y_j)$$

📌 Inference by Enumeration

14.

$$P(X \mid e) = \alpha \sum_y P(X, e, y)$$



📌 **Inference by Enumeration**

14.

$$P(X \mid e) = \alpha \sum_{y} P(X, e, y)$$

Where:

- X is the query variable
- e is the evidence
- y is over hidden variables
- α is a normalization constant

📌 **Joint Probability Computation (Bayesian Networks)**

15. Example:

$$P(light, no, delayed, miss) = P(light) \cdot P(no \mid light) \cdot P(delayed \mid light, no) \cdot P(miss \mid delayed)$$

📌 **Markov Assumption**

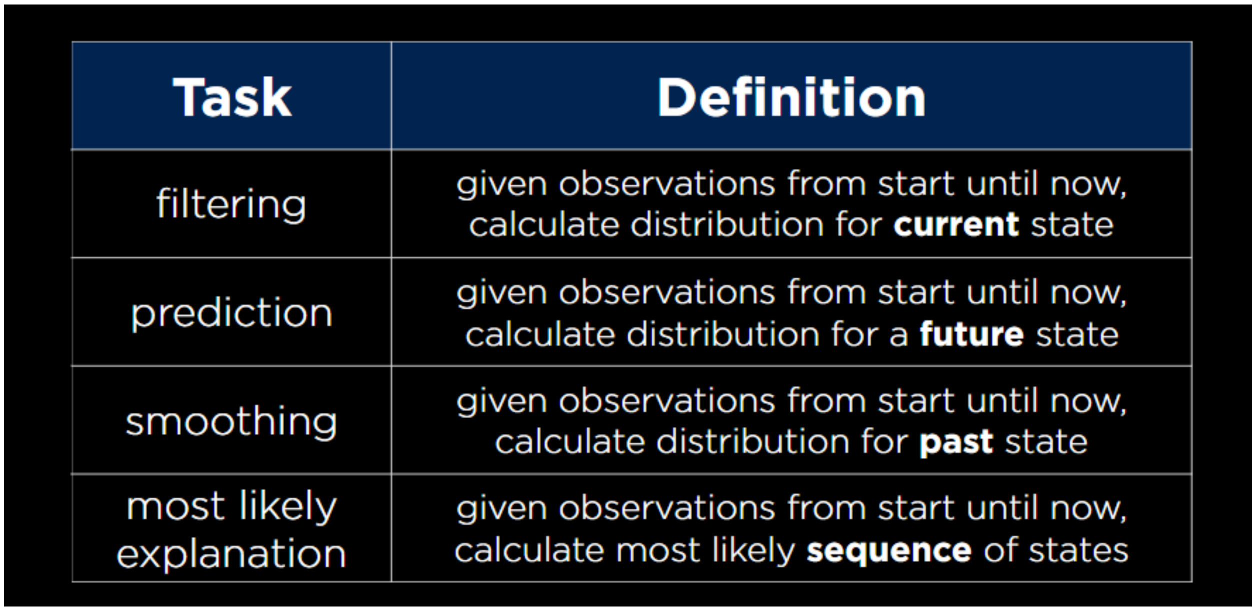
16.

$$P(X_{t+1} \mid X_t, X_{t-1}, \dots) = P(X_{t+1} \mid X_t)$$

📌 **Sensor Markov Assumption**

17.

$$P(E_t \mid X_t, X_{t-1}, \dots) = P(E_t \mid X_t)$$



Task	Definition
filtering	given observations from start until now, calculate distribution for **current** state
prediction	given observations from start until now, calculate distribution for a **future** state
smoothing	given observations from start until now, calculate distribution for **past** state
most likely explanation	given observations from start until now, calculate most likely **sequence** of states

ID	Requirement	Summary
NFR1	Scalability	Supports 10,000 users with modular design and Firebase database.
NFR2	Reliability	99.9% uptime with local processing for network-independent operation.
NFR3	Usability	Intuitive interface with <30-minute training for diverse users.
NFR4	Security	HIPAA-compliant data protection with encryption and MFA.
NFR5	Maintainability	Maintenance tasks completed in <15 minutes with plug-and-play design.

ID	Requirement	Summary
FR1	Hybrid Power System	LiFePO4 batteries and solar panels extend range beyond 30 km, enhancing autonomy and sustainability.
FR2	Modular Chassis	Aluminum chassis with tiered configurations ensures affordability and easy upgrades.
FR3	Hybrid Connectivity	Wi-Fi and BLE via ESP32 enable reliable operation in low-coverage areas.
FR4	AI Health Monitoring	AI predicts health emergencies with >98% accuracy, improving safety.
FR5	LiDAR Navigation	95% accurate autonomous navigation with LiDAR and SLAM for dynamic environments.
FR6	Multi-Modal Controls	EEG, joystick, and voice controls cater to diverse disabilities with high precision.
FR7	Plug-and-Play Maintenance	Modular wiring and app-based guides simplify maintenance for caregivers.
FR8	User Feedback System	App-based feedback loop drives continuous improvement via cloud analytics.