Knowledge Graphs

A **knowledge graph** is a structured representation of information that connects entities (such as people, places, concepts, or events) and their relationships in a way that mimics how humans organize and process knowledge. It's a network of nodes (entities) and edges (relationships) that is designed to provide meaningful context and insights from data.

Key Features of a Knowledge Graph:

Entities and Relationships:

Nodes represent entities (e.g., "Albert Einstein" or "Theory of Relativity"), and edges represent the relationships between them (e.g., "discovered by" or "related to").

Semantic Meaning:

Each entity and relationship has a clear definition, enabling the graph to understand and infer information.

Querying and Reasoning:

Allows for complex queries, such as "Who were the collaborators of Albert Einstein in developing theories?" or "What discoveries are related to quantum physics?"

Interconnected Data:

Combines information from multiple sources, linking diverse datasets to provide a unified perspective.

Applications:

•**Search Engines**: Google uses its Knowledge Graph to display information panels for direct answers to queries.

•**Recommender Systems**: For suggesting products, movies, or books based on interconnected user preferences and content data.

•**Healthcare**: For connecting symptoms, diseases, and treatments.

•**Enterprise Data Management**: Organizing corporate knowledge to improve decision-making and collaboration.

Examples:

Google Knowledge Graph: Provides summary cards on search results, e.g., displaying a quick fact box when you search for "Leonardo da Vinci.“

Wikidata: A free and open knowledge graph that serves as a structured data source for Wikipedia and other projects.

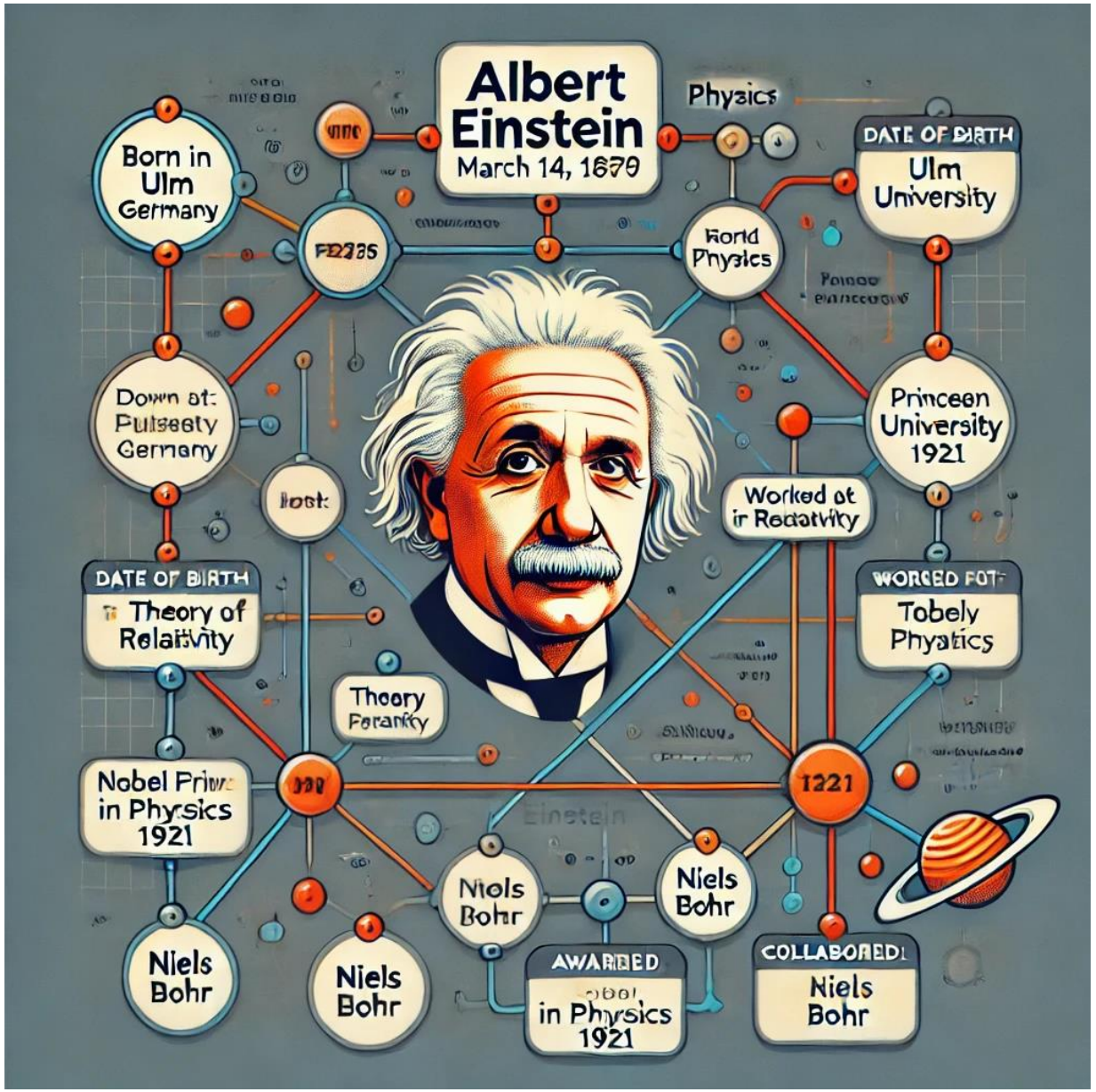
In essence, knowledge graphs help transform raw data into interconnected, actionable insights, making it easier to understand and utilize complex relationships.

Query: "Who is Albert Einstein?"

Entities and Relationships:

- **Albert Einstein** (Node)
  - **Born In** → Ulm, Germany (Node)
  - **Date of Birth** → March 14, 1879 (Attribute)
  - **Field** → Physics (Node)
  - **Known For** → Theory of Relativity (Node)
  - **Worked At** → Princeton University (Node)
  - **Awarded** → Nobel Prize in Physics (1921) (Node)
  - **Collaborated With** → Niels Bohr (Node)

Visual Representation:



Logical Agents & Planning

Logical agents are a type of artificial intelligence (AI) agent that make decisions and perform actions based on formal logic. These agents use **knowledge representation** and **logical reasoning** to deduce new facts, verify truths, and choose appropriate actions. Logical agents operate in domains where the rules, constraints, and environment can be described using formal logic (e.g., propositional logic or first-order logic).

Characteristics of Logical Agents:

Knowledge Representation:

They use logic to encode information about the world in the form of statements or formulas.

Reasoning:

Perform inference using rules of logic (e.g., modus ponens, resolution).

Decision Making:

Based on logical deductions, they decide the best course of action.

Soundness and Completeness:

1. **Soundness**: All derived conclusions are true if the premises are true.
2. **Completeness**: They can derive any truth from the given premises.

Components:

Knowledge Base (KB):

A repository of logical sentences that describe what the agent knows.

1. Example: *"If it rains, the ground will be wet"*.

Inference Mechanism:

Derives conclusions from the knowledge base.

1. Example: *"It is raining; therefore, the ground will be wet."*

Applications:

- **Game AI**:

  Use logic to predict opponents' moves.

- **Robotics**:

  Deduce safe paths or actions.

- **Automated Theorem Proving**:

  Solve mathematical or logical proofs.

Planning

Planning in AI is the process where an agent determines a sequence of actions that transitions it from its current state to a goal state. It involves reasoning about the future and often uses logic-based approaches to ensure actions are correct and efficient.

Logical Planning:

Logical agents can perform planning by encoding the problem in logical terms. This involves:

Initial State:

The starting condition of the agent or environment.

Goal State:

The desired outcome the agent needs to achieve.

Actions:

Defined in terms of preconditions (what must be true before the action) and effects (what becomes true after the action).

Example:

Scenario: A robot wants to deliver a package.

Initial State:

Robot is at Location A, and the package is at Location A.

Goal State:

Package is at Location B.

Actions:

1. Pick up the package (precondition: Robot and package are at the same location).
2. Move to Location B (precondition: Robot is at Location A).
3. Drop the package (precondition: Robot is at Location B with the package).

Assignment-3
Compare the planning approaches. Due Date: 29-11-24. Upload the pdf of your handwritten assignment on lms. Save the handwritten form with you.

Approaches to Planning:

Classical Planning:

Assumes a fully observable, deterministic world.
Example: STRIPS (Stanford Research Institute Problem Solver) formalism.

Heuristic Planning:

Uses heuristics to find efficient plans in large state spaces.

Hierarchical Planning:

Breaks down complex goals into sub-goals.

Probabilistic Planning: Handles uncertainty in actions or outcomes.

Integration of Logical Agents and Planning

Logical agents use their knowledge base and reasoning capabilities to perform planning:

Represent Actions and Goals in Logic:

Use predicates to define the domain.

Example: At(robot, A) ∧ At(package, A) (initial state).

Infer a Plan:

Deduce a sequence of actions to achieve the goal using logical inference.

Example: Deduce Move(robot, B) followed by Drop(package, B).

This combination is particularly effective in systems where actions must be carefully reasoned and justified, such as robotics, autonomous systems, or decision-support tools.

Rule-Based Knowledge Representation

Rule-based knowledge representation is a method of encoding knowledge in a system using **if-then rules** (also called production rules) to express logical relationships between facts and conclusions. It is widely used in expert systems, decision-making systems, and AI applications where reasoning is required.

Structure of Rule-Based Knowledge:

Rules:

- Represented as IF <condition> THEN <action/conclusion>.
- The **condition** specifies the facts or states that must be true.
- The **action/conclusion** specifies what to infer or do if the condition is satisfied.

  Example:

  IF it is raining THEN the ground is wet.

Facts:

- Represent known truths or observations about the world.

  Example:

  It is raining.

Inference Engine:

- Evaluates the rules against the facts to draw conclusions or decide actions.
- Uses techniques like **forward chaining** or **backward chaining** for reasoning.

Reasoning Mechanisms:

1.Forward Chaining (Data-Driven):

1. Starts with the known facts.
2. Applies rules to infer new facts or conclusions.
3. Proceeds step by step until a goal is reached.

Example:

Starting from "It is raining," deduce "The ground is wet.“

2.Backward Chaining (Goal-Driven):

1. Starts with the desired goal or conclusion.
2. Works backward to find rules and facts that support the goal.

Example:

To confirm "The ground is wet," check if "It is raining" is true.

Advantages of Rule-Based Systems:

1. **Simplicity**:
   Easy to understand and implement using logical structures.
2. **Modularity**:
   Rules are independent and can be added, removed, or modified without affecting others.
3. **Transparency**:
   Rules are explicit, making it easy to trace the reasoning process.

Limitations:

1. **Scalability**:
   As the number of rules grows, the system becomes harder to manage.
2. **Uncertainty Handling**:
   Cannot directly handle probabilistic or uncertain knowledge without extensions.
3. **Dependence on Experts**:
   Requires domain experts to manually encode all rules, which can be time-consuming.

Applications:

1.Expert Systems:

Medical diagnosis systems like MYCIN for identifying diseases.

2.Decision Support Systems:

Recommending loans in banking based on credit rules.

3.Business Automation:

Rule-based engines for approving insurance claims.

Example:

Scenario: Diagnosing a patient with a fever.

Fact: Patient has fever and rash.

Rules:

- IF patient has fever, THEN check for rash.
- IF patient has fever AND rash THEN suggest measles.

The inference engine uses forward chaining to suggest the diagnosis measles based on the rules and facts.

Enhancements to Rule-Based Systems:

To address limitations, additional techniques like **fuzzy logic**, **probabilistic rules**, or integration with machine learning can improve rule-based systems for more complex reasoning tasks.

Reasoning Under Uncertainty

Uncertainty

- Representation using first-order logic and propositional logic with certainty, which means we were sure about the predicates.
- With this knowledge representation, we might write P→Q, which means if P is true then Q is true
  - But consider a situation where we are not sure about whether
    - P is true or not then we cannot express this statement, this situation is called uncertainty

Reasons of Uncertainty

- Information occurred from unreliable sources.
- Experimental Errors
- Equipment fault
- Temperature variation
- Climate change.

What is Probabilistic Reasoning

- Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge.
- In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

- We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.
- In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behavior of someone for some situations," "A match between two teams or two players." These are probable sentences for which we can assume that it will happen but not sure about it, so here we use probabilistic reasoning.

Why Probabilistic Reasoning

- When there are unpredictable outcomes.
- When specifications or possibilities of predicates becomes too large to handle.
- When an unknown error occurs during an experiment

What is Probability

- Probability can be defined as a chance that an uncertain event will occur.
- It is the numerical measure of the likelihood that an event will occur.]
- The value of probability always remains between 0 and 1 that represent ideal uncertainties.

Probability in Uncertainty

- $0 \leq P(A) \leq 1$, where P(A) is the probability of an event A.
- P(A) = 0, indicates total uncertainty in an event A.
- P(A) =1, indicates total certainty in an event A.

$$\textbf{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

- P(¬A) = probability of a not happening event.
- P(¬A) + P(A) = 1.

Some Terms

- **Event:**
  - Each possible outcome of a variable is called an event.
- **Sample space:**
  - The collection of all possible events is called sample space.
- **Random variables:**
  - Random variables are used to represent the events and objects in the real world.
- **Prior probability:**
  - The prior probability of an event is probability computed before observing new information.
- **Posterior Probability:**
  - The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information.

Conditional Probability

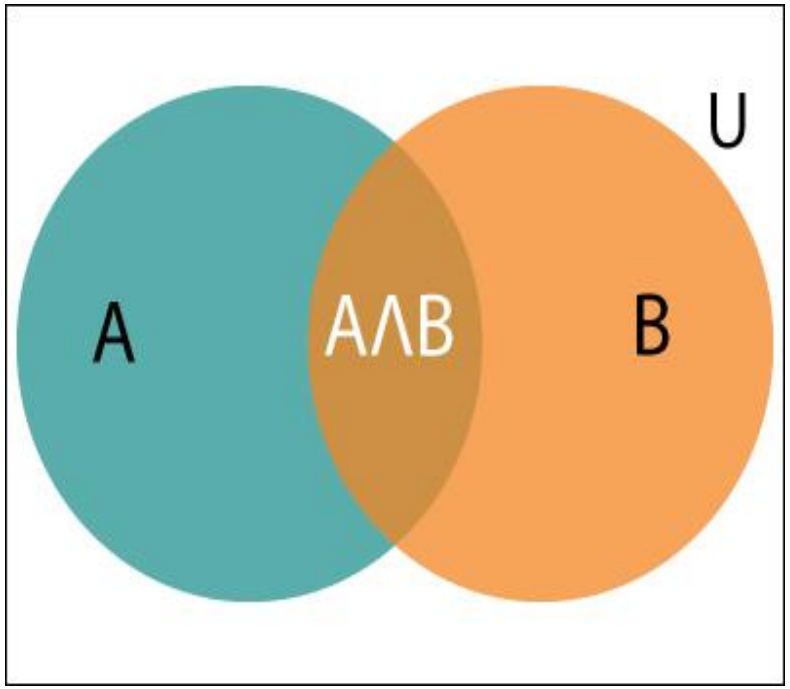
- Conditional probability is a probability of occurring an event when another event has already happened.

- Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:

- **Where P(*A*∧*B*)=** $P(A|B) = \frac{P(A \wedge B)}{P(B)}$ **f a and B**
- **P(B)= Marginal .** ,

- If the probability of A is given and we need to find the probability of B, then it will be given as:

$$P(B|A) = \frac{P(A \wedge B)}{P(A)}$$



Example

- **Example:**
  - In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?

- Let, A is an event that a student likes Mathematics
- B is an event that a student likes English.

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{0.4}{0.7} = 57\%$$

Bayes' Theorem

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge.
- It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes' Theorem

- **P(A|B) is Posterior probability**:
  - Probability of hypothesis A on the observed event B.
- **P(B|A) is Likelihood probability**:
  - Probability of the evidence given that the probability of a hypothesis is true.
- **P(A) is Prior Probability**:
  - Probability of hypothesis before observing the evidence.
- **P(B) is Marginal Probability**:
  - Probability of Evidence

Working of Naïve Bayes' Classifier

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability

Naive Bayes Classifier-Example for Discrete Data

To start with, let us consider a dataset.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit(“Yes”) or unfit(“No”) for playing golf.

Naive Bayes Classifier-Example for Discrete Data

Here is a tabular representation of our dataset.

	Outlook	Temperature	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No

	Outlook	Temperature	Humidity	Windy	Play Golf
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

Naive Bayes Classifier-Example for Discrete Data

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of dependent features. In above dataset, features are 'Outlook', 'Temperature', 'Humidity' and 'Windy'.
Response vector contains the value of **class variable(prediction or output)** for each row of feature matrix. In above dataset, **the class variable name is 'Play golf'**.

Assumption:

The fundamental Naive Bayes assumption is that each feature makes an:

- ✓ Independent
- ✓ Equal

contribution to the outcome.

Naive Bayes Classifier-Example for Discrete Data

With relation to our dataset, this concept can be understood as:

- We assume that no pair of features are dependent. For example, the temperature being ‘Hot’ has nothing to do with the humidity or the outlook being ‘Rainy’ has no effect on the winds. Hence, the features are assumed to be **independent**.
- Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can’t predict the outcome accurately. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

NOTE

The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Naive Bayes Classifier-Example for Discrete Data

Now, with regards to our dataset, we can apply Bayes' theorem in following way:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

where, y is class variable and X is a dependent feature vector (of size *n*) where:

$$X = (x_1, x_2, x_3, ..., x_n)$$

Just to clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of dataset)

```
X = (Rainy, Hot, High, False)
y = No
```

So basically, P(y|X) here means, the probability of "Not playing golf" given that the weather conditions are "Rainy outlook", "Temperature is hot", "high humidity" and "no wind".

Naive Bayes Classifier-Example for Discrete Data

Naive Assumption

The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Now, its time to put a naive assumption to the Bayes' theorem, which is, **independence** among the features. So now, we split **evidence** into the independent parts.

Now, if any two events A and B are independent, then,

$$P(A,B) = P(A)P(B)$$

Naive Bayes Classifier-Example for Discrete Data

Hence, we reach to the result:

$$P(y|x_1,...,x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

which can be expressed as:

$$P(y|x_1,...,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$$

Now, as the denominator remains constant for a given input, we can remove that term:

$$P(y|x_1,...,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

Naive Bayes Classifier-Example for Discrete Data

Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable *y* and pick up the output with maximum probability. This can be expressed mathematically as:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

So, finally, we are left with the task of calculating P(y) and $P(x_i \mid y)$.

Please note that P(y) is also called **class probability** and $P(x_i \mid y)$ is called **conditional probability**.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

Naive Bayes Classifier-Example for Discrete Data

Let us try to apply the above formula manually on our weather dataset. For this, we need to do some precomputations on our dataset.
We need to find $P(x_i \mid y_j)$ for each x_i in X and y_j in y. All these calculations have been demonstrated in the tables below:

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

Naive Bayes Classifier-Example for Discrete Data

So, in the figure above, we have calculated $P(x_i \mid y_j)$ for each x_i in X and y_j in y manually in the tables 1-4. For example, probability of playing golf given that the temperature is cool, i.e P(temp. = cool | play golf = Yes) = 3/9.

Also, we need to find class probabilities (P(y)) which has been calculated in the table 5. For example, P(play golf = Yes) = 9/14.

So now, we are done with our pre-computations and the classifier is ready!

Naive Bayes Classifier-Example for Discrete Data

Let us test it on a new set of features (let us call it today):

```
today = (Sunny, Hot, Normal, False)
```

So, probability of playing golf is given by:

$$P(Yes|today) = \frac{P(SunnyOutlook|Yes)P(HotTemperature|Yes)P(NormalHumidity|Yes)P(NoWind|Yes)P(Yes)}{P(today)}$$

and probability to not play golf is given by:

$$P(No|today) = \frac{P(SunnyOutlook|No)P(HotTemperature|No)P(NormalHumidity|No)P(NoWind|No)P(No)}{P(today)}$$

Naive Bayes Classifier-Example for Discrete Data

Since, P(today) is common in both probabilities, we can ignore P(today) and find proportional probabilities as:

$$P(Yes|today) \propto \frac{2}{9}.\frac{2}{9}.\frac{6}{9}.\frac{6}{9}.\frac{9}{14} \approx 0.0141$$

and

$$P(No|today) \propto \frac{3}{5}.\frac{2}{5}.\frac{1}{5}.\frac{2}{5}.\frac{5}{14} \approx 0.0068$$

Now, since

$$P(Yes|today) + P(No|today) = 1$$

Naive Bayes Classifier-Example for Discrete Data

These numbers can be converted into a probability by making the sum equal to 1 (normalization):

$$P(Yes|today) = \frac{0.0141}{0.0141+0.0068} = 0.67$$

and

$$P(No|today) = \frac{0.0068}{0.0141+0.0068} = 0.33$$

Since

$$P(Yes|today) > P(No|today)$$

So, prediction that golf would be played is 'Yes'.

Naive Bayes Classifier for Continuous Data

The method that we discussed above is applicable for discrete data. In case of continuous data, we need to make some assumptions regarding the distribution of values of each feature. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

In continuous case, we either discretize the continuous interval of x_i into bins $\{b_1, \ldots, b_m\}$ and proceed the same as discrete case, or we *assume* a function like Gaussian (or any other one), as follows:

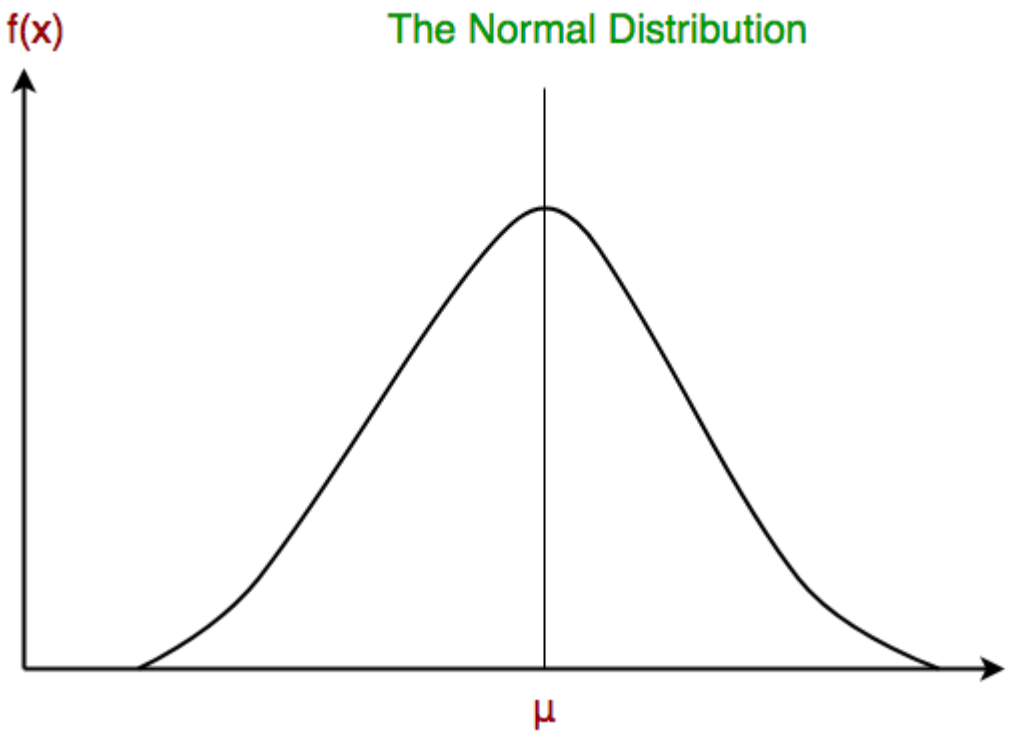
$$P(x_i \mid C_k) = \frac{1}{\sqrt{2\pi}\sigma_{i,k}} e^{-(x_i - \mu_{i,k})^2 / 2\sigma_{i,k}^2}$$

This way, for each feature-class pair (i, k), $P(x_i \mid C_k)$ is represented with two parameters $\{\mu_{i,k}, \sigma_{i,k}\}$ instead of a table in discrete case. The estimation of the parameters is the same as fitting a Gaussian distribution to one dimensional data, that is:

$$\hat{\mu}_{i,k} = \frac{\sum_{n:n \in C_k} x_{n,i}}{N_k}, \hat{\sigma}_{i,k}^2 = \frac{\sum_{n:n \in C_k} (x_{n,i} - \hat{\mu}_{i,k})^2}{N_k - 1}$$

Gaussian Naive Bayes Classifier

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**. A Gaussian distribution is also called **Normal distribution**. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:



The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

Gaussian Naive Bayes Classifier

The rest is independent of feature type

Representing $P(C_k)$ is the same for both discrete and continuous cases and is estimated as $\hat{P}(C_k) = N_k/N$.

Finally, the classifier is

$$C(x_n) = \underset{k \in \{1,..,K\}}{\operatorname{argmax}} \hat{P}(C_k) \prod_i \hat{P}(x_i = x_{n,i} | C_k)$$

Or equivalently using log-probabilities,

$$C(x_n) = \underset{k \in \{1,..,K\}}{\operatorname{argmax}} \log \hat{P}(C_k) + \sum_i \log \hat{P}(x_i = x_{n,i} | C_k)$$

Instead of Gaussian, we can opt for a more complex function, even a neural network. In that case, we should look for a technique to fit the function to data just like what we did with Gaussian.

Next Time:

- Markov Models
- Fuzzy Logic
- Inference using Fuzzy Rules