MACHINE LEARNING

Unit-5

Important Q & A

1) What are the Morkav Chain Monte Carlo (MCMC) methods?

Ans)

1. **Metropolis-Hastings Algorithm**: This is a basic MCMC method that involves sampling from a proposal distribution and accepting or rejecting these samples based on an acceptance criterion. It's very flexible and can be used for a wide range of problems.
2. **Gibbs Sampling**: A specialized MCMC method where you sample from the conditional distribution of each variable in turn, given the current values of the other variables. It's particularly useful for high-dimensional problems.
3. **Hamiltonian Monte Carlo (HMC)**: This method uses concepts from physics to propose samples. It involves calculating the gradient of the target distribution, which makes it more efficient for high-dimensional problems.
4. **Slice Sampling**: An adaptive method that involves sampling uniformly from the region under the probability density curve. It doesn't require tuning parameters and can handle complicated, multi-modal distributions.

Each of these methods has its strengths and weaknesses, and the choice of which to use depends on the specific problem and the properties of the target distribution.

2) Given a brief explanation about Monte Carlo Sampling and Markov Chains with diagrams?

Ans) Sampling

Sampling is a way to approximately estimate certain characteristics of the whole population by taking a subset of the population into the study.

Sampling has various use cases —

- It could be used to approximate an intractable sum or integral.
- It could be used to provide a significant speedup in estimating tractable but costly sum or integral.
- In some cases like density estimation, it could simply be used to approximate probability distribution and then impute missing data.

Few Sampling techniques — **Ancestral Sampling, Inverse Transform Sampling, Rejection Sampling, Importance Sampling, Monte Carlo Sampling, MCMC Sampling.**

In this post, we are solely focusing on the MCMC method, other methods are a discussion for another post.

Introduction

MCMC methods are a family of algorithms that uses Markov Chains to perform Monte Carlo estimate.

The name gives us a hint, that it is composed of two components —
Monte Carlo and *Markov Chain.* Let us understand them separately and in their combined form.

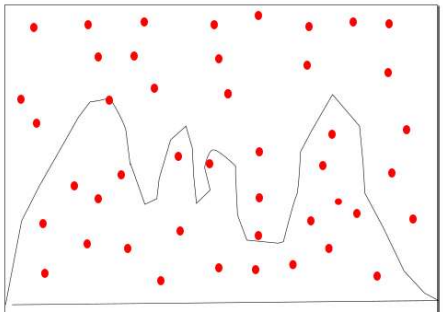
Monte Carlo Sampling

(Intuitively)

Monte Carlo method derives its name from a Monte Carlo casino in Monaco.

It is a technique for sampling from a probability distribution and using those samples to approximate desired quantity. In other words, it uses randomness to estimate some deterministic quantity of interest.

Example: If we are asked to calculate the area under the curve for the given curve in the below image, it might require integrating over complex analytical formula.
However using the Monte Carlo method, we will randomly generate red dots (more dots for more accuracy) in the rectangle and calculate the ratio of dots falling under the curve w.r.t dots falling in the entire rectangle — the ratio will provide us with the area, given the area of the rectangle.



(Image by author) — Monte Carlo to estimate the area under the curve

Basically, if calculating some quantity has a complex analytical structure, we can simply perform a simulation to generate lots of samples and use them to approximate the quantity. These works provided asymptotically they follow central limit theorem.

It has many other use cases in risk analysis, reliability analysis etc.

(Mathematically)

Say we have Expectation (s) to estimate, this could be a highly complex integral or even intractable to estimate— using the Monte Carlo method we resort to approximate such quantities by averaging over samples.

$$s = \int p(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} = E_p[f(\mathbf{x})]$$

Original Expectation to be calculated

$$\hat{s}_n = \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{x}^{(i)})$$

Approximated Expectation generated by stimulating large samples of f(x)

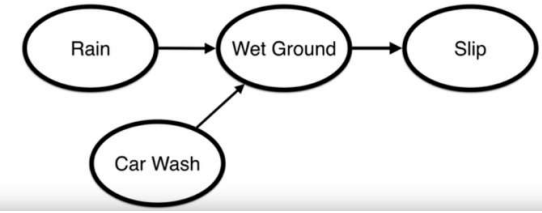
Computing the average over a large number of samples could reduce the standard error and provide us with a fairly accurate approximation.

"***This method has a limitation, for it assumes to easily sample from a probability distribution, however doing so is not always possible. Sometimes, we can't even sample from the distribution. In such cases, we make use of Markov chains to efficiently sample from an intractable probability distribution.***"

Markov Chains

Before getting into ***Markov chains***, let us have a look over useful property that defines it —

Markov property :



Entities in the Oval shapes are states

Consider a system of 4 states we have from the above image—

'*Rain'* or '*Car Wash'* causing the '*Wet Ground'* followed by *'Wet Ground'* causing the *'Slip'.*

Markov property simply makes an assumption — ***the probability of jumping from one state to the next state depends only on the current state and not on the sequence of previous states that lead to this current state.***

If we are to calculate the probability of someone slipping, knowing whether the ground is wet or not provides enough evidence to estimate it. We need not require the states that lead to it ('Rain' or 'Car Wash').

mathematically speaking :

$$P(X_{n+1} = k | X_n = k_n, X_{n-1} = k_{n-1}, \ldots, X_1 = k_1) = P(X_{n+1} = k | X_n = k_n)$$

Markov property truncating the distribution

It is evident from the mathematical equation that the Markov property assumption could potentially save us a lot of computation.

In hindsight, If a process exhibits ***Markov Property***, then it is known as ***Markov Chain.***

Now that we have seen Markov Chain, let us discuss the property that makes it so desirable — **Stationary Distribution.**

Stationary Distribution :

Suppose, we have a process of few states and we have a fixed transition probability **(Q)** of jumping between states.
We start with some random probability distribution over all states ($\mathbf{S_i}$)
at time step **i**, and to estimate the probability distribution over all states at the next time step i.e **i+1**, we multiply it by transition probability **Q**.

$$s_{i+1} = s_i Q$$

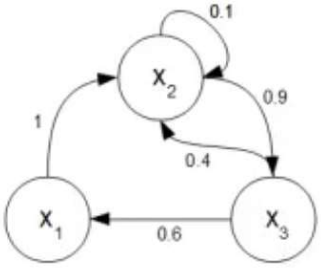
If we keep on doing this, after a while **S** stops changing on multiplying with matrix Q, this is when we say it has reached a ***stationary distribution.***

$$sQ = s$$

stationary distribution has been reached

Let us see this with an example —

In this example, we have 3 states (X_1, X_2, X_3)



$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \end{bmatrix}$$

Transition Probability between states (**T**)

If we are in the state $\mathbf{S_2}$, the probability of staying put in $\mathbf{S_2}$ is 0.1,
transitioning to state $\mathbf{S_1}$ is 0, and transitioning to state $\mathbf{S_3}$ is 0.9 (as evident from the second row in the matrix).

Let us start with some random value of vector $\mathbf{S_i}$ (vector shows the probability of being at each state at any particular time step), we can see how the vector sums up to 1.

$$S_i = [\ 0.5,\ 0.2,\ 0.3\ ]$$

$$S_{i+1} = [\ 0.18,\ 0.64,\ 0.18\ ]$$

after applying $S_{i+1}=S_i*Q$

If we keep moving in time steps, eventually we will reach the stationary state,

$$[\ 0.22,\ 0.40,\ 0.38\ ]$$

stationary distribution

now the best part to know is this stationary distribution does not depend on the initial state, you can try experimenting with various initial state S_i.

3) Explain about Proposal Probability or Proposal distribution?

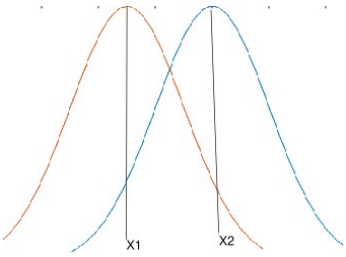
Ans) Metropolis — Hasting Algorithm

Suppose we are sampling from distribution p(x) = f(x) / Z, where Z is the intractable normalization constant.

Our objective is to sample from p(x) in such a way that involves making use of numerator alone and avoids having to estimate denominator.

(Proposal Probability)

Let us start by looking at the **proposal probability (g)**.
Given a sample, it proposes us with the next potential sample in the Markov chain.
(How to decide whether to accept or reject the potential sample is something we'll see in the next section).



(Image taken by author) — Proposal Distribution

Assume that $\mathbf{g(X_2 \mid X_1) = Normal(\ X_1,\ \sigma\ )}$
(*it could have been any distribution, for simplicity we chose normal distribution****)***

Keeping X_1 as mean, we make a normal distribution. Then we sample X_2 from this distribution.

We repeat the same step for sampling X_3, by keeping X_2 as the mean.

(Main Algorithm)

Let us begin this algorithm by following the detailed balance sheet condition.

$$p(X_1).T(X_1 \rightarrow X_2) = p(X_2).T(X_2 \rightarrow X_1)$$

Detailed Balance Sheet

The probability of transitioning from X_1 to X_2 can be seen as a two-step process, considering we are at state X_1 —

The first step is to make a proposal of state X_2 with some ***proposal probability*** **g** (discussed in the previous section).

The second step is to accept the new state X_2 with some ***acceptance probability*** **A.**

Substituting the transition probability into the equation…

$$p(X_1).g(X_1|X_2).A(X_1 \rightarrow X_2) = p(X_2).g(X_2|X_1).A(X_2 \rightarrow X_1)$$

Substituting the p(x) with f(x)/Z, being on both sides Z gets cancelled out, we get…

$$f(X_1).g(X_1|X_2).A(X_1 \rightarrow X_2) = f(X_2).g(X_2|X_1).A(X_2 \rightarrow X_1)$$

restructuring the equation leads to

$$\frac{A(X_1 \rightarrow X_2)}{A(X_2 \rightarrow X_1)} = \frac{f(X_2)}{f(X_1)} \frac{g(X_2 \mid X_1)}{g(X_1 \mid X_2)}$$

Substituting short-hand notation

$$\frac{f(X_2)}{f(X_1)} \rightarrow R_f \qquad \frac{g(X_2 \mid X_1)}{g(X_1 \mid X_2)} \rightarrow R_g$$

$$\frac{A(X_1 \rightarrow X_2)}{A(X_2 \rightarrow X_1)} = R_f R_g$$

finally, we get the **acceptance probability A**

$$A(X_1 \rightarrow X_2) = max(1, R_f R_g)$$

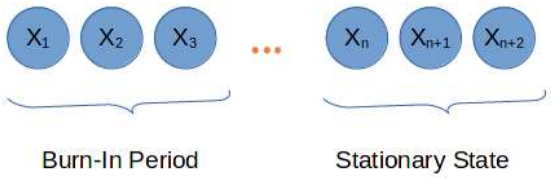
Acceptance Probability

Summarize —

- we start with a random state.

- based on proposal probability (g) we randomly pick a new state.
- calculate the acceptance probability (A) of the proposed new state.
- flip coin with probability to land head is equal to acceptance probability, if the coin lands head accept the sample else reject it.
- repeat the process for quite some time.

We keep on sampling for a long time and discard the initial few samples as the chain has not reached its stationary state yet (this period is known as
burn-in period).



Limitation:

- After from chain getting stuck while approximating multi-modal distribution and getting biased samples and therefore less accurate estimate of the desired quantity.
- Metropolis-Hasting gets very slow when the sample space is high-dimension. (another amazing MCMC method **Hamiltonian Monte Carlo** overcomes these short-comings and is a discussion for another post).

4) What are Morkav Chain Monte Carlo (MCMC) Graphical Models?

Ans) Graphical models in machine learning provide a powerful framework for representing complex relationships among variables. They are particularly useful for modeling dependencies and conditional independencies, which can simplify the representation of joint probability distributions.

Types of Graphical Models

Graphical models can be broadly categorized into two types: directed and undirected models.

Directed Graphical Models (Bayesian Networks)

- **Definition**: These models represent variables as nodes and dependencies as directed edges. Each node is associated with a probability distribution that quantifies the effect of its parent nodes.
- **Applications**: Commonly used in decision-making processes, causal inference, and reasoning under uncertainty.

Undirected Graphical Models (Markov Random Fields)

- **Definition**: In these models, the relationships between variables are represented by undirected edges. They are particularly useful for capturing symmetric relationships.
- **Applications**: Often used in image processing, spatial data analysis, and social network analysis.

Advantages of Graphical Models

- **Modularity**: They allow for the decomposition of complex problems into simpler, manageable components.
- **Interpretability**: The graphical structure provides intuitive insights into the relationships between variables.
- **Scalability**: Graphical models can efficiently handle large datasets by exploiting the structure of the data.

Implementation Considerations

When implementing graphical models, it is crucial to consider the following aspects:

- **Data Representation**: Choose an appropriate data structure to represent the graph, such as adjacency lists or matrices, depending on the sparsity of the graph.
- **Inference Algorithms**: Utilize algorithms like belief propagation or Markov Chain Monte Carlo (MCMC) for inference tasks.
- **Learning Parameters**: Employ techniques such as maximum likelihood estimation or Bayesian inference to learn the parameters of the model from data.

5) What is Bayesian Networks explain its types and structure?

Ans) Bayesian networks, also known as belief networks or Bayesian belief networks (BBNs), are powerful tools for representing and reasoning about uncertain knowledge. These networks use a graphical structure to encode probabilistic relationships among variables, making them invaluable in fields such as artificial intelligence, bioinformatics, and decision analysis.

Basic Structure of Bayesian Networks

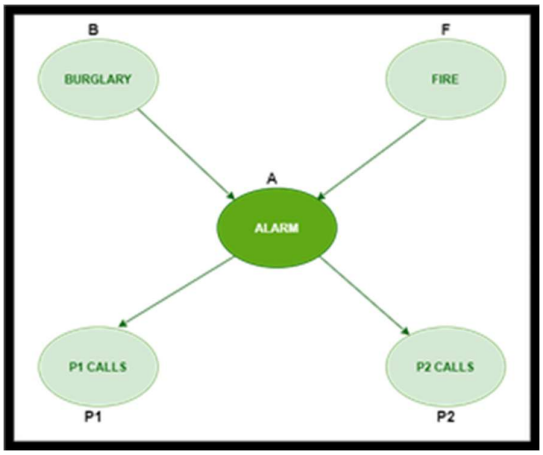
A Bayesian network consists of:

- **Nodes**: Each node represents a random variable, which can be discrete or continuous.
- **Edges**: Directed edges (arrows) between nodes represent conditional dependencies.

Bayesian Belief Network is a graphical representation of different probabilistic relationships among random variables in a particular set. It is a classifier with no dependency on attributes i.e it is condition independent. Due to its feature of joint probability, the probability in Bayesian Belief Network is derived, based on a condition — P(attribute/parent) i.e probability of an attribute, true over parent attribute.

(Note: A classifier assigns data in a collection to desired categories.)

- Consider this example:



- In the above figure, we have an alarm 'A' – a node, say installed in a house of a person 'gfg', which rings upon two probabilities i.e burglary 'B' and fire 'F', which are – parent nodes of the alarm node. The alarm is the parent node of two probabilities P1 calls 'P1' & P2 calls 'P2' person nodes.
- Upon the instance of burglary and fire, 'P1' and 'P2' call person 'gfg', respectively. But, there are few drawbacks in this case, as sometimes 'P1' may forget to call the person 'gfg', even after hearing the alarm, as he has a tendency to forget things, quick. Similarly, 'P2', sometimes fails to call the person 'gfg', as he is only able to hear the alarm, from a certain distance.

For example, if node A influences node B, there would be a directed edge from A to B, indicating that B is conditionally dependent on A.

Conditional Independence

The fundamental property of Bayesian networks is that they encode conditional independence relationships between variables. This means that each node is conditionally independent of its non-descendants given its parents. This property significantly reduces the complexity of the network by breaking down the joint probability distribution into simpler, local distributions.

Joint Probability Distribution

A Bayesian network defines a joint probability distribution over its variables. The joint probability of a set of variables can be expressed as the product of the conditional probabilities of each variable given its parents:

If we have variables x1, x2, x3,....., xn, then the probabilities of a different combination of x1, x2, x3.. xn, are known as Joint probability distribution.

$P[x_1, x_2, x_3, ..., x_n]$, it can be written as the following way in terms of the joint probability distribution.

= P[x_1| x_2, x_3,....., x_n]P[x_2, x_3,....., x_n]

= P[x_1| x_2, x_3,....., x_n]P[x_2|x_3,....., x_n]....P[x_{n-1}|x_n]P[x_n].

In general for each variable Xi, we can write the equation as:

$P(X_i|X_{i-1},\ldots\ldots\ldots, X_1) = P(X_i \mid Parents(X_i))$

Inference in Bayesian Networks

Inference in Bayesian networks involves computing the probability distribution of a subset of variables given known values for other variables. This can be achieved through various methods:

- **Exact Inference**: Algorithms like Variable Elimination and Junction Tree Algorithm.
- **Approximate Inference**: Techniques like Monte Carlo methods and Loopy Belief Propagation.

These inference methods are crucial for querying the network and making predictions based on observed data.

Learning Bayesian Networks

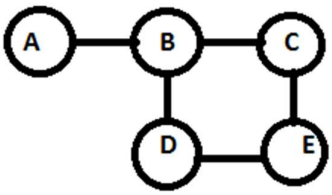
Learning a Bayesian network involves two main tasks:

- **Structure Learning**: Determining the network structure (i.e., the DAG).
- **Parameter Learning**: Estimating the conditional probability distributions.

6) What is Markov Random Field Model?

Ans) The Markov Random Field model is a model which use an undirected graph. Undirected graphical models edge represents the potential between two variables, syntactically, Factorization distribution probabilities between variable. In each Individual variable connected with the edge represent a certain clique in the graph; means probability distribution of the variables in the graph can factorize an individual clique potential function.

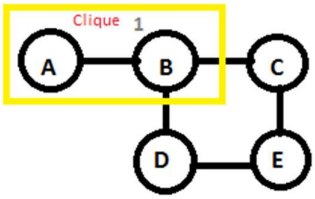
Just as we had CPDs for Bayesian networks, we have tables to incorporate relations between nodes in **Markov networks**. However, there are two crucial differences between these tables and CPDs.



.

Clique in graph theory.it is a subset of vertices of an undirected graph.

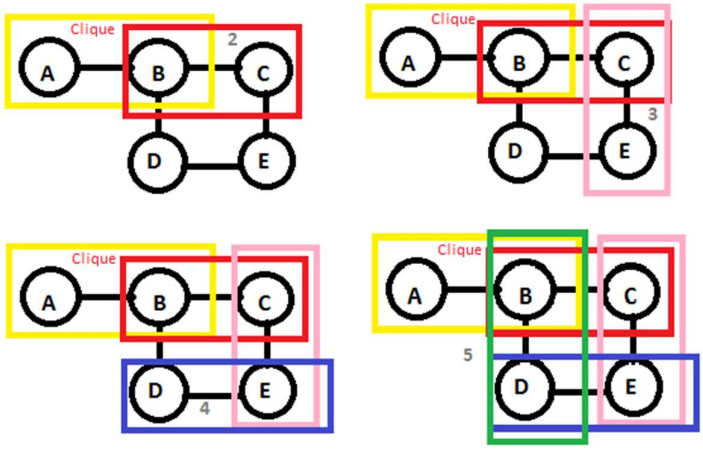
P(A, B, C, D, E) α Φ(A,B) Φ (B,C) Φ (B,D) Φ (C,E) Φ (D,E)



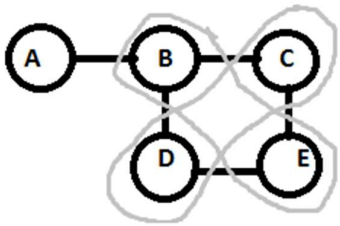
$$P(X) = \frac{1}{z} \bar{\Pi}_{Ceclique} \phi_C(x_C) \quad \text{(Potential functions)}$$

Such that: It induces sub graph is complete in every vertices in a clique is adjacent. So, clique in this graph adjust adjacently one by one.

Like,

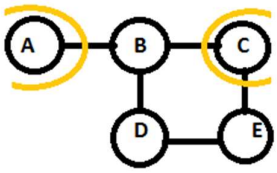


It is some different if we join **D,C** and **B,E** clique over here then it is also change its probability.



P(A, B, C, D, E) α Φ(A,B) Φ (B,C,D) Φ (C,D,E)

Some undirected graphic model has Markov Random Field. In MRF certain paths between **A** and **C**.



A -> B -> C

A -> B -> D -> E -> C

Note that: Markov networks do not need to be acyclic, as was the case with Bayesian networks.

Independence properties such as Markov properties:

- Any 2 subsets if variables are conditionally independent given a separating subset.
- If we take **'A'** as a subset and **'C'** as one subset then there is wore between them. So, there is no way to go between **'A'** and **'C'** without getting threw the subset. So, we are using **(A, B)** than **B, C, D, E**.

Therefore, A and C are separating subsets

- Any 2 subset of variable are conditionally independent given a separating subset.
- {B,D}, {B,E} and {B,D,E} are separating subsets.

7) What are Hidden Markov Models? Explain its key concepts & applications?

Ans) Hidden Markov Models (HMMs) are statistical models used to represent systems that transition between hidden states over time, with each state producing observable outputs. HMMs are widely used in various applications such as speech recognition, bioinformatics, and finance

Key Concepts of Hidden Markov Models

Here are the main Key Concepts of Hidden Markov Models.

1. **Hidden States**: The system is assumed to be in one of several hidden states at any given time. These states are not directly observable.
2. **Observable Outputs**: Each hidden state generates observable outputs (emissions) according to a probability distribution.
3. **Transition Probabilities**: The probability of transitioning from one hidden state to another.
4. **Emission Probabilities**: The probability of observing a particular output given a hidden state.
5. **Initial State Probabilities**: The probability distribution over the initial hidden state.

Model Components for Hidden Markov Models

Here are the main Model Components for Hidden Markov Models.

1. **States**: A finite set of states (e.g., S = {S1, S2, ..., Sn}).
2. **Observations**: A finite set of possible observations (e.g., O = {O1, O2, ..., Om}).
3. **Transition Matrix**: The matrix A where A[i, j] represents the probability of transitioning from state i to state j.
4. **Emission Matrix**: The matrix B where B[i, k] represents the probability of observing output k given state i.
5. **Initial State Distribution**: The vector π where π[i] is the probability of starting in state i.

Applications of Hidden Markov Models

- **Speech Recognition**: Modeling sequences of spoken words.
- **Bioinformatics**: Gene prediction and protein sequence analysis.
- **Finance**: Modeling stock prices and market trends.

8) What are Tracking Methods?

Ans) Key Tracking Methods

1. Manual Logging:

- **Description**: Keeping a record of experiments manually in spreadsheets, notebooks, or documents.
- **Pros**: Simple and flexible.
- **Cons**: Prone to human error, difficult to scale, and lacks automation.

2. Automated Logging with Tools:

- **Description**: Using software tools to automatically log parameters, metrics, artifacts, and versions.
- **Pros**: Reduces human error, enhances reproducibility, and provides easy access to historical data.
- **Cons**: Requires learning and integrating new tools into the workflow.

3. Version Control Systems (VCS):

- **Description**: Using Git or other VCS to track code, configuration files, and sometimes data.
- **Pros**: Supports collaboration, branching, and history tracking.

- **Cons**: Requires manual management of large data files.

4. Cloud-Based Experiment Tracking:

- **Description**: Utilizing cloud platforms like AWS SageMaker, Azure ML, or Google Cloud AI for experiment tracking.
- **Pros**: Scalability, integration with cloud resources, and access to cloud storage and compute power.
- **Cons**: Costs can accumulate, requires familiarity with cloud services.

5. Specialized Experiment Tracking Platforms:

- **Description**: Dedicated platforms like MLflow, DVC, ClearML, and Comet.
- **Pros**: Tailored features for machine learning, ease of use, and integration with other tools.
- **Cons**: Learning curve, potential cost for advanced features.

Popular Tools for Tracking

1. **MLflow:** An open-source platform for managing the ML lifecycle, including experiment tracking, model registry, and deployment.
2. **DVC (Data Version Control):** A Git-based tool for versioning data, models, and pipelines.
3. **TensorBoard:** A visualization tool for TensorFlow that helps track and visualize experiments.
4. **Comet:** A managed solution for experiment tracking with dashboards and reports.
5. **ClearML:** A comprehensive tool for tracking experiments, automating ML operations, and creating pipelines.

Best Practices for Tracking

- **Consistency**: Use a consistent naming convention and structure for experiments.
- **Automation**: Automate as much of the logging process as possible to reduce errors.
- **Documentation**: Keep detailed notes on the purpose, changes, and outcomes of each experiment.
- **Visualization**: Use dashboards and visual tools to monitor and compare experiments.