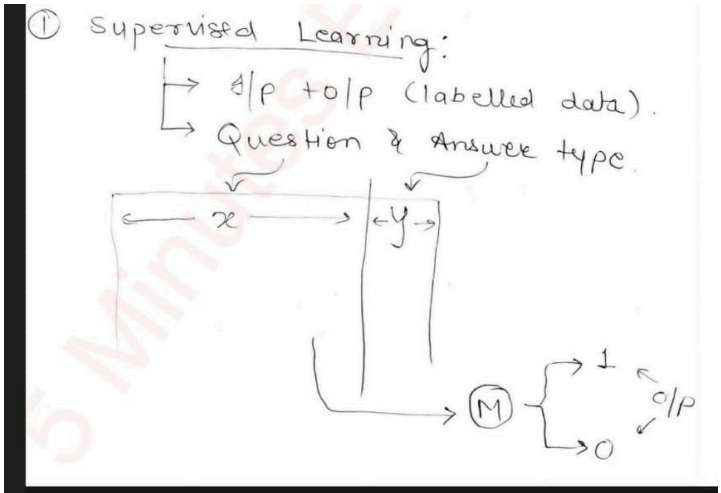
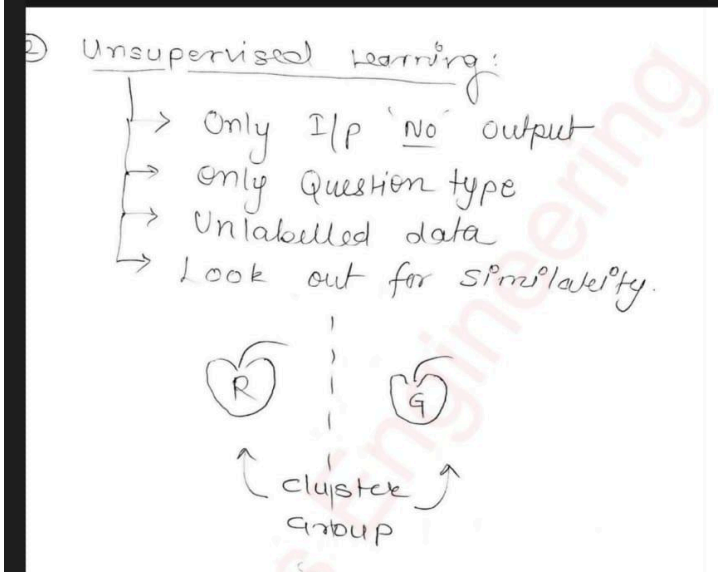
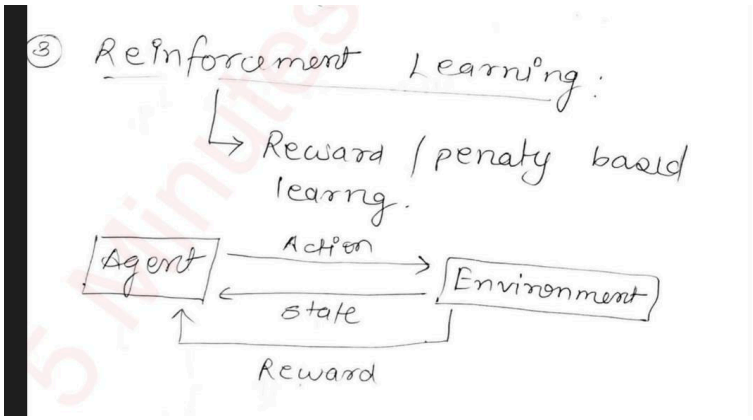
Machine Learning is a way for computers to learn and make decisions from data without being explicitly programmed.







Classic Machines:

1. **Definition**: Machines that operate based on predefined instructions or rules written by humans.
2. **Characteristics**:
    - Fixed functionality.
    - Cannot learn or improve over time.
    - Requires reprogramming to change behavior.
3. **Examples**:
    - Calculators.
    - Basic washing machines with fixed wash cycles.
    - Rule-based automation systems.

Adaptive Machines:

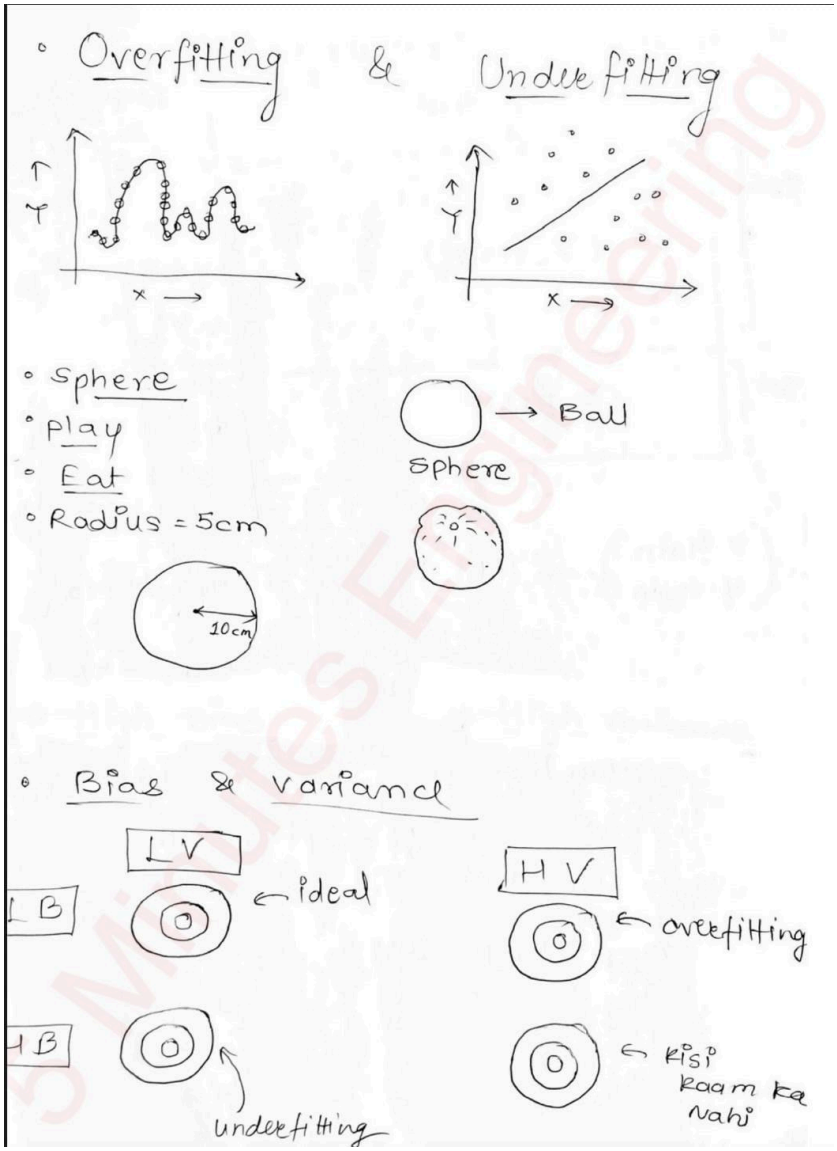
1. **Definition**: Machines that use algorithms and data to learn, adapt, and improve their performance over time without explicit reprogramming.
2. **Characteristics**:
    - Self-learning capabilities.
    - Can handle changing environments or tasks.
    - Often based on Machine Learning or AI techniques.
3. **Examples**:
    - Self-driving cars that learn from driving data.
    - Personalized recommendation systems like Netflix or Amazon.
    - AI chatbots that adapt to user interactions.

Labeled data is data that comes with answers or tags, like pictures labeled as "cat" or "dog," which helps machines learn what they are.

A **cost function** is a mathematical formula used in machine learning to measure how far a model's predictions are from the actual results. It helps the model learn by showing how much error it made, so it can improve during training.

Simple Example:

- If a model predicts a price of $50 but the actual price is $40, the cost function calculates the error (e.g., 10 in this case).
- The goal is to minimize this error.



Overfitting:

- **Definition**: When a machine learning model learns too much from the training data, including noise and irrelevant details, making it perform poorly on new, unseen data.
- **Key Issue**: High accuracy on training data but poor generalization to test or real-world data.

- **Example**: A model memorizes every detail of training examples instead of understanding the underlying pattern.

Underfitting:

- **Definition**: When a model fails to learn enough from the training data, resulting in poor performance on both training and test data.
- **Key Issue**: Too simple to capture the patterns in the data.
- **Example**: Using a straight line to fit complex, curved data.

Key Difference:

- **Overfitting**: Model is too complex.
- **Underfitting**: Model is too simple.

Bias:

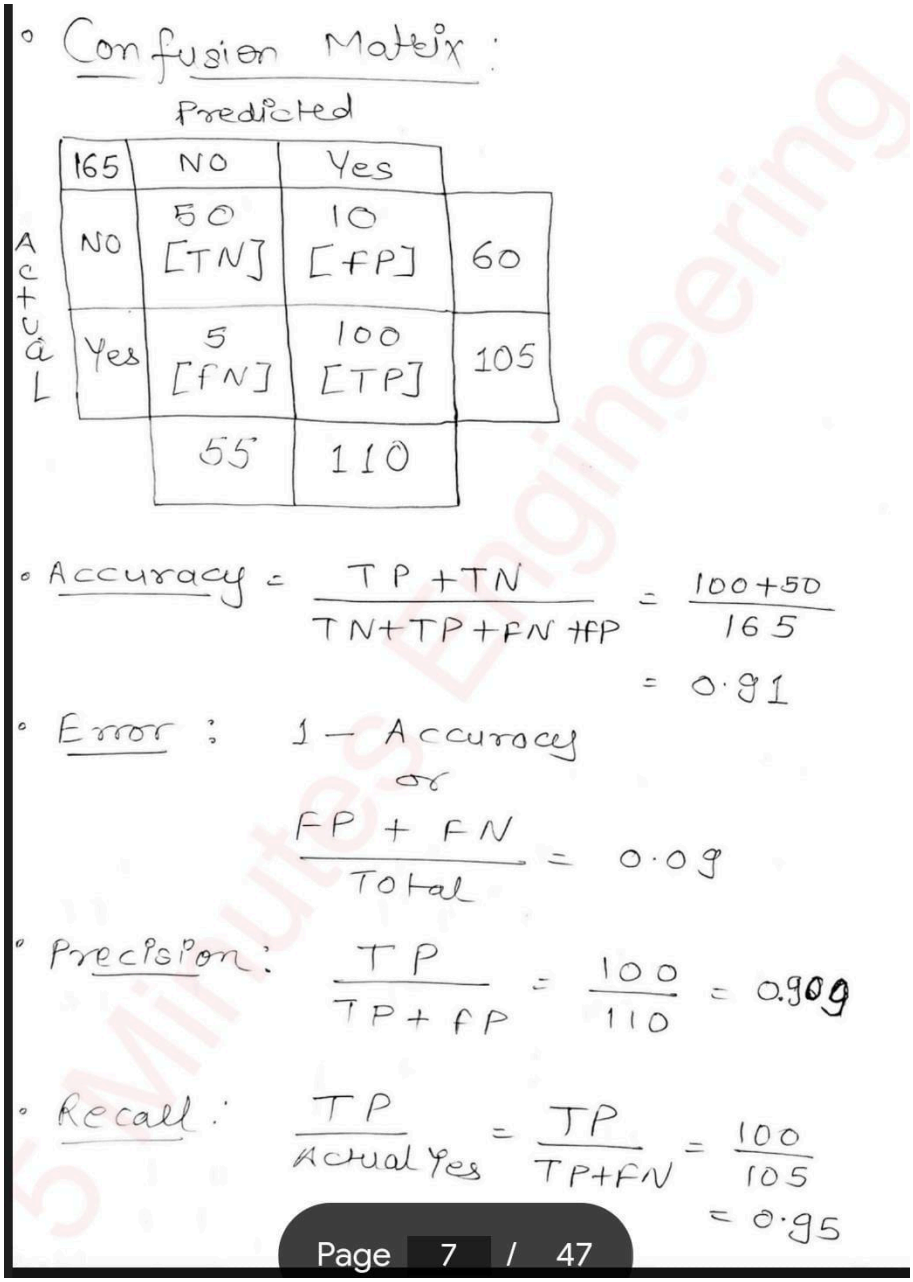
- **Definition**: Bias refers to the error introduced by simplifying a model too much, leading to inaccurate predictions even on training data.
- **Key Issue**: The model makes assumptions that limit its ability to capture the true patterns.
- **Example**: Using a straight line to predict house prices in a complex market.
- **Relation**: High bias often leads to underfitting.

Variance:

- **Definition**: Variance refers to the error introduced by a model being too sensitive to small fluctuations in the training data, causing it to perform poorly on new data.
- **Key Issue**: The model learns noise and details that don't generalize well.
- **Example**: A model predicts perfectly for training data but performs poorly on test data.
- **Relation**: High variance often leads to overfitting.

Key Difference:

- **Bias**: Too simple; misses important patterns (underfitting).
- **Variance**: Too complex; learns noise (overfitting).



A **confusion matrix** is a table used to evaluate the performance of a classification model by comparing its predictions with the actual outcomes. It provides detailed insights into the types of errors the model makes.

Structure:

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Terms Explained:

1. **True Positive (TP)**: Correctly predicted positive cases.
2. **True Negative (TN)**: Correctly predicted negative cases.
3. **False Positive (FP)**: Incorrectly predicted as positive (Type I error).
4. **False Negative (FN)**: Incorrectly predicted as negative (Type II error).

Metrics Derived:

1. **Accuracy** = (TP+TN)/(TP+TN+FP+FN)(TP + TN) / (TP + TN + FP + FN)(TP+TN)/(TP+TN+FP+FN)
2. **Precision** = TP/(TP+FP)TP / (TP + FP)TP/(TP+FP)
3. **Recall (Sensitivity)** = TP/(TP+FN)TP / (TP + FN)TP/(TP+FN)
4. **F1 Score** = 2×(Precision×Recall)/(Precision+Recall)2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})2×(Precision×Recall)/(Precision+Recall)

Curse of Dimensionality refers to the problems and challenges that arise when working with data that has a large number of features (dimensions). As the number of dimensions increases, the volume of the data space grows exponentially, making it difficult for machine learning algorithms to effectively analyze and learn from the data.

Key Issues:

1. **Sparse Data**: In high-dimensional spaces, data points become sparse, making it hard to find patterns or meaningful insights.
2. **Increased Complexity**: The algorithms require more computational power and time as the dimensions grow.
3. **Overfitting**: With more dimensions, models may start to memorize the data rather than generalize, leading to overfitting.
4. **Distance Metrics**: In high-dimensional spaces, the concept of distance (e.g., Euclidean distance) becomes less meaningful, causing issues in clustering or nearest neighbor algorithms.

Example:

Imagine a dataset with 2 features. You can visualize it in a 2D plane, where you can easily see patterns. But if you add 100 more features, the data points spread out, and finding meaningful relationships becomes much harder.

To tackle this, dimensionality reduction techniques (like PCA) are often used to reduce the number of features while preserving important information.

Preprocessing is the process of preparing and cleaning raw data before feeding it into a machine learning model. It helps improve the model's performance by transforming the data into a suitable format, removing noise, and handling inconsistencies.

Normalization: When features have different units or ranges and need to be scaled to the same range (0 to 1).
Standardization: When features have a Gaussian distribution or when your model assumes that.
Robust Scaler: When your data has outliers, and you want to scale features without letting outliers dominate.

Normalization: Scales data to a range of 0 to 1. Useful when features have different ranges or units.
Formula: x−minmax−min\frac{x - \text{min}}{\text{max} - \text{min}}max−minx−min
Standardization (Scaling): Rescales data to have a mean of 0 and a standard deviation of 1. Useful when data follows a Gaussian distribution.
Formula: x−μσ\frac{x - \mu}{\sigma}σx−μ
Robust Scaler: Scales data using the median and interquartile range (IQR), making it resistant to outliers.
Formula: x−medianIQR\frac{x - \text{median}}{\text{IQR}}IQRx−median

Label Encoding is a technique used to convert categorical data into numeric values. Each unique category is assigned a specific integer label. This is particularly useful when the categorical data is ordinal (i.e., there is a meaningful order to the categories).

Example:

For a feature "Color" with categories `["Red", "Green", "Blue"]`, label encoding would convert them as:

- Red → 0
- Green → 1
- Blue → 2

One-Hot Encoding is a technique used to convert categorical data into binary (0 or 1) columns. Each category is represented by a separate binary column, where only one column is "1" (indicating the presence of that category) and all others are "0."

Example:

For a feature "Color" with categories `["Red", "Green", "Blue"]`, one-hot encoding would create 3 new columns:

- Red → [1, 0, 0]
- Green → [0, 1, 0]
- Blue → [0, 0, 1]