Machine Learning Study Notes (Session 6–35)

Session 6: Introduction to ML / AI

1. Artificial Intelligence (AI)

Artificial Intelligence is the broad field of creating systems that can **perform tasks that normally require human intelligence**.

Core abilities of AI

- Learning from data
- Reasoning & decision-making
- Problem solving
- Perception (vision, speech)
- Language understanding

Types of AI

- **Narrow AI (Weak AI)**
  Performs a specific task
  👉 Example: Face recognition, ChatGPT, recommendation systems
- **General AI (Strong AI)**
  Human-level intelligence across tasks (still theoretical)

2. Machine Learning (ML)

Machine Learning is a **subset of AI** where systems learn **patterns from data instead of being explicitly programmed**.

ML Pipeline

1. Collect data
2. Preprocess data
3. Choose model
4. Train model
5. Evaluate performance
6. Deploy & monitor

For example, voice assistants like Siri understanding speech commands. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 7: Python Environment Setup

A Python environment is an isolated setup that includes the **Python interpreter**, **installed libraries**, and **dependencies** required to run a program. In machine learning and AI projects, different projects often need different versions of libraries like NumPy, Pandas, TensorFlow, or PyTorch. Using environments prevents version conflicts and ensures that projects run consistently across different systems.

For example, setting up isolated Python environments for different ML projects. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 8: Python Basics

1. Python Syntax

Python syntax refers to the **rules that define how Python programs are written and structured**. One of Python's most important features is its use of **indentation instead of braces** to define code blocks. This makes Python code clean, readable, and easy to understand. Statements end without semicolons, and code readability is enforced by design, reducing logical errors.

2. Data Types

Data types specify the **kind of data a variable can store**. Python supports several built-in data types such as integers (int), floating-point numbers (float), strings (str), and boolean values (bool). Python is dynamically typed, meaning the type of a variable is determined at runtime, which allows flexible and fast development.

3. Type Conversion

Type conversion refers to **changing one data type into another**. Python supports implicit conversion (automatic) and explicit conversion using functions like int(), float(), and str(). This is particularly useful when dealing with user inputs or data loaded from files, which are often in string format.

For example, writing a Python script to calculate student grades. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 9: Flow Control

1. Flow Control

Flow control refers to the **order in which statements in a program are executed**. In Python, flow control is mainly handled using conditional statements such as if, elif, and else. These statements allow a program to make decisions based on conditions and execute different blocks of code accordingly.

2. Loops

Loops are used to **repeat a block of code multiple times**. Python provides for loops for iterating over sequences and while loops for repeated execution based on a condition. Loops are fundamental in automating repetitive tasks and handling large datasets efficiently.

For example, using loops to iterate through sensor data readings. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 10: Object Oriented Programming

1. What is OOP?

Object-Oriented Programming is a way of writing programs by **grouping data and functions together**. Instead of thinking only in steps, you think in terms of **objects**, just like real life (student, car, bank account).

In Python, OOP helps you organize code better, especially when programs become large. It makes code easier to understand, reuse, and manage.

For example, designing a class structure for a banking application. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 11: NumPy & Pandas

1. What is NumPy?

NumPy (Numerical Python) is a Python library used for **working with numbers and arrays**. It allows fast mathematical operations on large amounts of data. The main feature of NumPy is the **array**, which is more powerful and faster than Python lists.

In machine learning, NumPy is used for numerical computations like matrix operations, vector calculations, and handling large datasets efficiently. Almost all ML libraries internally use NumPy for fast calculations.

4. What is Pandas?

Pandas is a Python library used for **data handling and analysis**. It is built on top of NumPy and makes working with structured data easy. Pandas is mainly used for reading, cleaning, manipulating, and analyzing datasets.

In ML projects, Pandas is used to load datasets (CSV, Excel), handle missing values, filter data, and prepare data before feeding it into models.

For example, analyzing COVID-19 case data using Pandas. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 12: Data Visualization

1. What is Data Visualization?

Data Visualization is the process of **representing data in graphical form** such as charts, graphs, and plots. Instead of reading large tables of numbers, visualization helps us **see patterns, trends, and relationships** easily.

In data science and machine learning, visualization is used to understand data before training models and to analyze results after training. Good visualizations help in making better decisions and explaining insights clearly.

For example, visualizing monthly sales trends using line charts. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 13: Scikit-Learn

1. What is Scikit-Learn?

Scikit-Learn is a popular Python library used for **machine learning**. It provides ready-to-use tools for tasks like classification, regression, clustering, and model evaluation. It is built on top of **NumPy, Pandas, and Matplotlib**, making it fast and easy to use.

In ML projects, Scikit-Learn is widely used because it offers simple syntax and consistent structure. With just a few lines of code, you can train models, make predictions, and evaluate performance.

2. Why Use Scikit-Learn?

Scikit-Learn makes machine learning **easy and efficient**, especially for beginners. It removes the need to write algorithms from scratch and allows developers to focus on understanding data and results.

For students and professionals, Scikit-Learn is ideal for experimenting with different algorithms, comparing models, and building prototypes. It is commonly used in academics, industry projects, and interviews.

For example, training a spam email classifier using scikit-learn. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 14: Statistics

1. What is Statistics?

Statistics is the branch of mathematics used to **collect, analyze, interpret, and present data**. It helps us understand data by summarizing large datasets into meaningful numbers and patterns.

In machine learning and data science, statistics is the foundation. Models learn patterns from data, and statistics helps us understand whether those patterns are meaningful or just random.

For example, calculating average income and variance in census data. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 16: Hypothesis Testing

1. What is Hypothesis Testing?

Hypothesis testing is a statistical method used to **make decisions using data**. It helps determine whether there is enough evidence in a sample to support or reject an assumption about a population.

In data science and machine learning, hypothesis testing is used to verify claims, compare models, and check whether observed results are meaningful or happened by chance.

---

2. Statistical Hypotheses

A hypothesis is a statement about a population. There are two types:

- **Null hypothesis (H_0)**: Assumes no effect or no difference
- **Alternative hypothesis (H_1 or H_a)**: Assumes there is an effect or difference

In ML, a null hypothesis might say that a new model performs the same as an old one, while the alternative hypothesis claims the new model performs better.

---

3. Level of Significance (α)

The **level of significance**, denoted by α, is the probability of rejecting the null hypothesis when it is actually true. Common values are **0.05** or **0.01**.

In practical terms, α sets a threshold for decision-making. In ML experiments, it controls how confident we want to be before claiming that a model improvement is real.

For example, testing whether a new drug improves recovery rate. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 17: Exploratory Data Analysis (EDA)

1. What is Exploratory Data Analysis (EDA)?

Exploratory Data Analysis (EDA) is the process of **understanding a dataset before applying machine learning models**. It involves summarizing data, visualizing patterns, and identifying issues such as missing values or outliers.

In ML projects, EDA is one of the most important steps. A good understanding of data helps in choosing the right preprocessing techniques and models, leading to better predictions and performance.

---

2. Objectives of EDA

The main goal of EDA is to **get familiar with the data**. This includes understanding the structure, types of variables, and relationships between features.

In machine learning, EDA helps answer questions like: What features are important? Is the data balanced? Are there unusual patterns? These insights guide the entire ML workflow.

---

3. Understanding the Dataset

EDA starts with understanding the **size, shape, and structure** of the dataset. This includes checking the number of rows and columns, data types, and basic information.

In ML, this step helps identify categorical and numerical features, which is essential for preprocessing steps like encoding and scaling.

For example, exploring customer churn data before model building. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 18: Data Cleaning

1. What is Data Cleaning?

Data Cleaning is the process of **fixing or removing incorrect, incomplete, or inconsistent data** from a dataset. Real-world data is rarely perfect and often contains missing values, errors, duplicates, or irrelevant information.

In machine learning, clean data is extremely important because models learn directly from data. Poor-quality data leads to poor model performance, even if advanced algorithms are used.

---

2. Importance of Data Cleaning

Data cleaning improves **data quality and reliability**. Clean data helps models learn correct patterns and produce accurate predictions.

In ML projects, data cleaning is usually done after EDA and before model training. A well-cleaned dataset reduces noise, bias, and errors in the learning process.

---

3. Handling Missing Values

Missing values occur when data is not recorded properly. Data cleaning involves identifying missing values and deciding how to handle them.

Common techniques include removing rows, filling values with mean, median, or mode, or using more advanced imputation methods. Proper handling prevents errors during model training.

---

4. Removing Duplicate Data

Duplicate records can distort analysis and model results. Data cleaning includes detecting and removing duplicate rows from the dataset.

In ML, duplicates can cause models to overfit or give biased results. Removing duplicates ensures that each data point contributes fairly to learning.

For example, handling missing values in weather datasets. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 19: Encoding & Scaling

1. What is Encoding?

Encoding is the process of **converting categorical (text) data into numerical form** so that machine learning models can understand it. Most ML algorithms work only with numbers, so text labels must be transformed before training.

In real-world datasets, features like gender, city, or category are common. Encoding makes these features usable by models without changing their meaning.

---

2. Why Encoding is Needed

Machine learning models perform mathematical operations, which cannot be applied directly to text. Encoding ensures that categorical information is represented numerically.

Without encoding, models cannot process categorical features. Proper encoding helps models learn meaningful patterns from data.

For example, encoding categorical product categories for prediction models. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 20: Feature Engineering

1. What is Feature Engineering?

Feature Engineering is the process of creating, selecting, and transforming input features so that machine learning models can learn better patterns from data. Features are the input variables used by a model to make predictions.

In ML, good features are often more important than complex algorithms. Well-engineered features can significantly improve model accuracy and performance.

---

2. Why Feature Engineering is Important

Raw data is rarely in the best form for machine learning. Feature engineering helps convert raw data into meaningful and useful inputs for models.

In real-world ML projects, improving features often gives better results than simply changing the model. This makes feature engineering a key skill for data scientists.

For example, creating interaction features from housing price data. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 21: Feature Selection

1. What is Feature Selection?

Feature Selection is the process of **choosing the most important input features** for a machine learning model and removing irrelevant or redundant ones. Not all features in a dataset help the model; some may add noise.

In ML, feature selection helps models focus on useful information, improving performance and making learning more efficient.

---

2. Why Feature Selection is Important

Too many features can make a model complex and slow. Some features may be highly correlated or completely useless.

Feature selection helps reduce overfitting, improves accuracy, lowers computation cost, and makes models easier to interpret.

For example, selecting important medical test features for diagnosis. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 22: Preprocessing Workflow

1. What is a Preprocessing Workflow?

A **preprocessing workflow** is a **step-by-step process** used to prepare raw data before training a machine learning model. Real-world data is messy, so it must be cleaned, transformed, and organized properly.

In ML, preprocessing ensures that data is in a format that models can understand and learn from. A good workflow leads to better accuracy and stable model performance.

---

2. Why Preprocessing is Important

Machine learning models are sensitive to data quality. If preprocessing is poor, even the best algorithm will give bad results.

A proper workflow reduces errors, removes noise, and ensures consistency between training and testing data. It also makes ML experiments reproducible and reliable.

For example, building a preprocessing pipeline for an ML project. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 23: Regression

1. What is Regression?

Regression is a machine learning technique used to **predict continuous numerical values**. It finds the relationship between input features (independent variables) and a target value (dependent variable).

In ML, regression is commonly used for tasks like predicting house prices, salary estimation, temperature forecasting, and sales prediction.

---

2. How Regression Works

Regression models learn patterns by fitting a line or curve that best represents the relationship between inputs and output.

The model uses training data to learn this relationship and then predicts values for new, unseen data. The goal is to minimize prediction error.

For example, predicting house prices using linear regression. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 24: Regularization

1. What is Regularization?

Regularization is a technique used in machine learning to **prevent overfitting**. It helps the model perform well not only on training data but also on new, unseen data.

It works by **penalizing large model weights**, forcing the model to stay simple and avoid learning noise.

---

2. Why Regularization is Needed

Sometimes a model fits the training data too perfectly, capturing unnecessary details. This causes poor performance on test data.

Regularization adds a constraint so the model focuses on the most important patterns instead of memorizing data.

---

3. How Regularization Works

Regularization modifies the **loss (cost) function** by adding a penalty term based on model coefficients.

This penalty discourages overly complex models and reduces the risk of overfitting.

For example, reducing overfitting in regression using Ridge regression. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 25: Classification

1. What is Classification?

Classification is a machine learning technique used to **predict categories or class labels** instead of numerical values.

The output is a **discrete class**, such as yes/no, spam/not spam, or disease/no disease.

---

2. How Classification Works

A classification model learns patterns from labeled training data and assigns new data points to one of the predefined classes.

The goal is to correctly map input features to the correct class based on learned boundaries.

---

3. Types of Classification

Common types of classification include:

- **Binary Classification** (two classes)

- **Multiclass Classification** (more than two classes)
- **Multi-label Classification** (multiple labels at once)

Each type is used based on the problem.

For example, classifying emails as spam or non-spam. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 26: Tree Models

1. What are Tree Models?

Tree models are machine learning algorithms that make decisions by splitting data into branches, similar to a **flowchart**.

Each split is based on a feature, and the final decision is made at a **leaf node**.

---

2. How Tree Models Work

The model starts at the root node and repeatedly splits the data based on conditions like **greater than or less than** a value.

The goal is to create pure groups where data points mostly belong to one class or have similar values.

---

3. Decision Tree

A **Decision Tree** is the most basic tree model.

It is easy to understand and interpret, making it popular for both classification and regression tasks.

For example, using decision trees to approve loan applications. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 27: Ensemble Learning

1. What is Ensemble Learning?

Ensemble Learning is a machine learning technique where **multiple models are combined** to make better predictions than a single model.

The main idea is: *many weak models together form a strong model*.

---

2. Why Ensemble Learning is Used

A single model may make mistakes or overfit the data.

By combining multiple models, ensemble methods:

- Reduce errors
- Improve accuracy
- Increase model stability

---

3. How Ensemble Learning Works

Different models are trained on the same or different parts of the data.

Their predictions are then combined using methods like **voting** (classification) or **averaging** (regression).

For example, combining multiple models for credit risk prediction. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 28: Model Evaluation

1. What is Model Evaluation?

Model evaluation is the process of checking how well a machine learning model performs on **new, unseen data**.

It helps us understand whether the model is accurate, reliable, and ready for real-world use.

---

2. Why Model Evaluation is Important

A model may perform well on training data but poorly on test data.

Evaluation ensures the model:

- Generalizes well
- Does not overfit or underfit

- Makes correct predictions

---

3. Training, Validation, and Test Data

Data is usually split into:

- **Training set** – used to train the model
- **Validation set** – used to tune parameters
- **Test set** – used to evaluate final performance

This gives an unbiased performance estimate.

For example, evaluating classifiers using ROC-AUC score. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 31: Supervised ML Assessment

1. What is Supervised ML Assessment?

Supervised ML Assessment is the process of **checking and validating supervised machine learning models** to ensure they work correctly and give reliable predictions.

It focuses on evaluating models trained using **labeled data**.

---

2. Purpose of Supervised ML Assessment

The main goal is to confirm that the model:

- Learns correct patterns
- Performs well on unseen data
- Meets problem requirements

This step decides whether the model is ready for real use.

---

3. Types of Supervised Learning Models

Supervised learning includes:

- **Regression models** (predict numbers)

- **Classification models** (predict classes)

Assessment methods differ slightly for each type.

For example, developing an end-to-end student performance prediction system. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 32: Clustering

1. What is Clustering?

Clustering is an **unsupervised machine learning** technique used to group similar data points together **without using labels**.

Data points in the same cluster are more similar to each other than to those in other clusters.

---

2. Why Clustering is Used

Clustering helps discover **hidden patterns** in data when output labels are not available.

It is useful for data exploration and understanding structure in large datasets.

---

3. How Clustering Works

Clustering algorithms measure similarity using distance metrics like **Euclidean distance**.

Based on this similarity, data points are grouped into clusters.

For example, segmenting customers based on purchasing behavior. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 33: Dimensionality Reduction

1. What is Dimensionality Reduction?

Dimensionality Reduction is a technique used to **reduce the number of features** in a dataset while keeping as much important information as possible.

It helps simplify data and make models faster and more efficient.

---

2. Why Dimensionality Reduction is Needed

Datasets with many features can be hard to process and visualize.

Reducing dimensions helps:

- Improve model performance
- Reduce computation time
- Avoid overfitting
- Make data easier to understand

---

3. How Dimensionality Reduction Works

The technique transforms original features into a **smaller set of new features**.

These new features represent the most important patterns in the data.

For example, reducing image dimensions using PCA. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 34: Generative vs Discriminative Models

1. What are Generative and Discriminative Models?

In machine learning, models are broadly divided into **generative** and **discriminative** models based on **how they learn from data**.

Both are used mainly for **classification**, but they work differently.

---

2. Generative Models

Generative models learn the **joint probability** of input features and output labels.

They try to understand **how the data is generated**, allowing them to create new data samples.

---

3. How Generative Models Work

Generative models learn:

- The distribution of each class
- How features are related within a class

They can generate new data points similar to the training data.

For example, comparing Naive Bayes and Logistic Regression for text classification. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.

Session 35: Unsupervised Case Study

1. What is an Unsupervised Case Study?

An **unsupervised case study** applies unsupervised learning techniques to a **real-world dataset without labels**.

The goal is to discover patterns, groups, or structures in the data.

---

2. Objective of the Case Study

The main objectives are to:

- Understand the dataset
- Apply unsupervised algorithms
- Interpret results meaningfully

There is no "right answer," only insights.

---

3. Dataset Understanding

The first step is exploring the dataset to:

- Identify features
- Check data types
- Understand scale and distributions

This helps choose the right technique.

---

4. Data Preprocessing

Before applying algorithms, data is:

- Cleaned (missing values, outliers)
- Scaled or normalized
- Transformed if needed

Good preprocessing improves results.

---

5. Algorithm Selection

Common unsupervised algorithms used:

- K-Means
- Hierarchical Clustering
- DBSCAN
- PCA (for visualization)

Choice depends on data structure.

---

6. Applying the Model

The chosen algorithm is applied to the dataset.

The model groups data points based on similarity or reduces dimensions.

---

7. Evaluating Unsupervised Results

Since labels are not available, evaluation uses:

- Silhouette score
- Inertia (for K-Means)
- Visual inspection (plots)

Interpretation is key.

---

8. Result Interpretation

Clusters or components are analyzed to:

- Understand group characteristics
- Identify patterns or anomalies
- Extract business or domain insights

For example, clustering geographic data to identify crime hotspots. This demonstrates how the concepts discussed in this session are applied in real-life situations to solve practical problems using data-driven approaches.