Machine Learning Development Life Cycle (MLDLC)

→ MLDLC is a **set of guidelines** that must be followed when creating any machine learning based software product (from idea to end product).

Important Steps of MLDLC

Step 1: Framing Problem

→ This is the initial decision-making phase
→ It is the responsibility of the developer to frame the problem correctly from the start.
→ **Questions answered in this step include:** What is the exact problem to be solved? Who are the customers? What is the estimated cost? How large a team is required? What should the final product look like?
→ Decisions are made here regarding the type of machine learning required (Supervised or Unsupervised), the deployment mode (online or batch), necessary algorithms, and the data source.

Step 2: Data Gathering

→ Machine learning is impossible without data, and for company projects, data is often specific and not easily available.
→ **Data can be sourced from various points:**

* Direct CSV files.
* **APIs:** Writing Python code to hit an API, receive data (often in JSON format), and convert it into a preferred format (usually CSV).
* **Web Scraping:** Extracting publicly available data from websites
* **Databases (via Data Warehouses):** Direct ML models are generally not run on live transactional databases due to risks. Data is typically moved via **ETL (Extract, Transform, Load)** processes into a Data Warehouse, where ML tasks are performed.
* **Big Data Tools:** Data stored in tools like Hadoop or Cassandra (in HDFS) must be fetched using specific classes.

→ **Goal:** Gathering the necessary data and storing it in the correct format.

Step 3: Data Processing

→ Data obtained from various sources is often **unclean** or **dirty** and cannot be used directly, as this results in poor model accuracy.
→ **Common data issues:** Duplicates, missing data, errors, outliers, mismatched data from multiple sources, and differences in the number of columns.
→ Data Processing is synonymous with **Preprocessing** (changes made before training).
→ **Preprocessing tasks include:**

* Removing duplicates.
* Removing missing values and errors.

* **Scaling values:** Necessary when input columns have values of vastly different magnitudes (e.g., one value is large, another is decimal) to prevent algorithms that calculate distance from malfunctioning. This often involves techniques like normalization or Z-score normalization.

→ **Goal:** Bringing the data into a format that the machine learning algorithm can easily consume.

Step 4: Exploratory Data Analysis (EDA)

→ EDA involves analyzing the data to understand the relationship between input and output.
→ Thorough understanding of the data is required before a good model can be built.
→ **Tasks in EDA:**

* Performing experiments and discovering hidden relationships in the data.
* Plotting graphs and creating visualizations.
* **Univariate Analysis:** Independent analysis of each column (e.g., finding the mean or standard deviation).
* **Bivariate and Multivariate Analysis:** Studying the relationships between two or more columns.
* Handling **imbalanced datasets** (where one class has significantly more samples than others, like having many images of cats but few of dogs).

→ Spending more time on EDA allows you to gain a concrete understanding of the data, which helps in future decision-making.

Step 5: Feature Engineering and Selection

→ Features refer to the **input columns**, which are crucial as the output depends on them.
→ **Feature Engineering:** Creating intelligent new columns or making changes to existing features to simplify the analysis.
→ **Feature Selection:** Choosing only the necessary input columns.

* This is done because not every input feature influences the output.
* Reducing the number of columns also reduces the time required to train the model.

Step 6 & 7: Model Training, Evaluation, and Selection

→ Once the data is cleaned and features are selected, the model is ready for training.
→ **Model Training:** Different machine learning algorithms (from various families, such as deep learning, linear, or tree-based) are brought in and trained using the data.

* It is generally recommended to train multiple algorithms, as the best-performing algorithm depends on the specific data.

→ **Evaluation:** All trained models are evaluated using **performance metrics** (like accuracy, Mean Squared Error, or Gini Index) to determine how well they perform.

→ **Model Selection and Tuning:** One or more models are selected, and their parameters are fine-tuned.

* **Hyperparameter Tuning:** Adjusting the internal settings of the final model to further improve performance.

→ **Ensemble Learning:** Often, multiple machine learning algorithms are combined using techniques like Bagging, Boosting, or Stacking to create a single, more powerful model, which typically improves overall performance.
→ **Result:** A powerful machine learning model capable of making predictions.

Step 8: Deployment and Testing

→ The machine learning model must be converted into a software format (website, mobile app, desktop app) so users can interact with it.
→ **Deployment Process:**

* The model is converted into a **binary file** (e.g., using the Pickle tool).
* This file is converted into an **API**.
* When a user inputs data via a form, the input hits the server, is processed by a Python application, sent to the API, and the API uses the binary file to generate and return a prediction as JSON data.
* The model is hosted on a server (like Heroku, AWS, or GCP) to serve user requests.

→ **Testing (Beta Testing):** After the model is live, testing is conducted, often starting with a phased rollout to loyal customers who can provide reliable feedback.

* If the model is unsatisfactory, the process must loop back to earlier stages (e.g., data processing or feature selection) to fix the issue.

Step 9: Optimization

→ This final step involves perfecting and automating the entire process.
→ **Tasks in Optimization:**

* Launching the model to all customers.
* Setting up safety measures, including taking **backups** of both the model and the data.
* Handling potential crashes by setting up automation to **roll back** to a previous version if necessary.
* Managing **Load Balancing** to efficiently handle high volumes of user traffic and serve requests quickly.
* Determining the necessary **retraining frequency** (e.g., weekly or monthly).
* **Handling Data Drift (R.O.W.I.N.G.):** Over time, a model's performance degrades as the nature of the data changes (e.g., a mask detection system failing when new types of masks appear). Retraining is necessary to combat this and must be automated.
* Optimizing costs throughout the process.