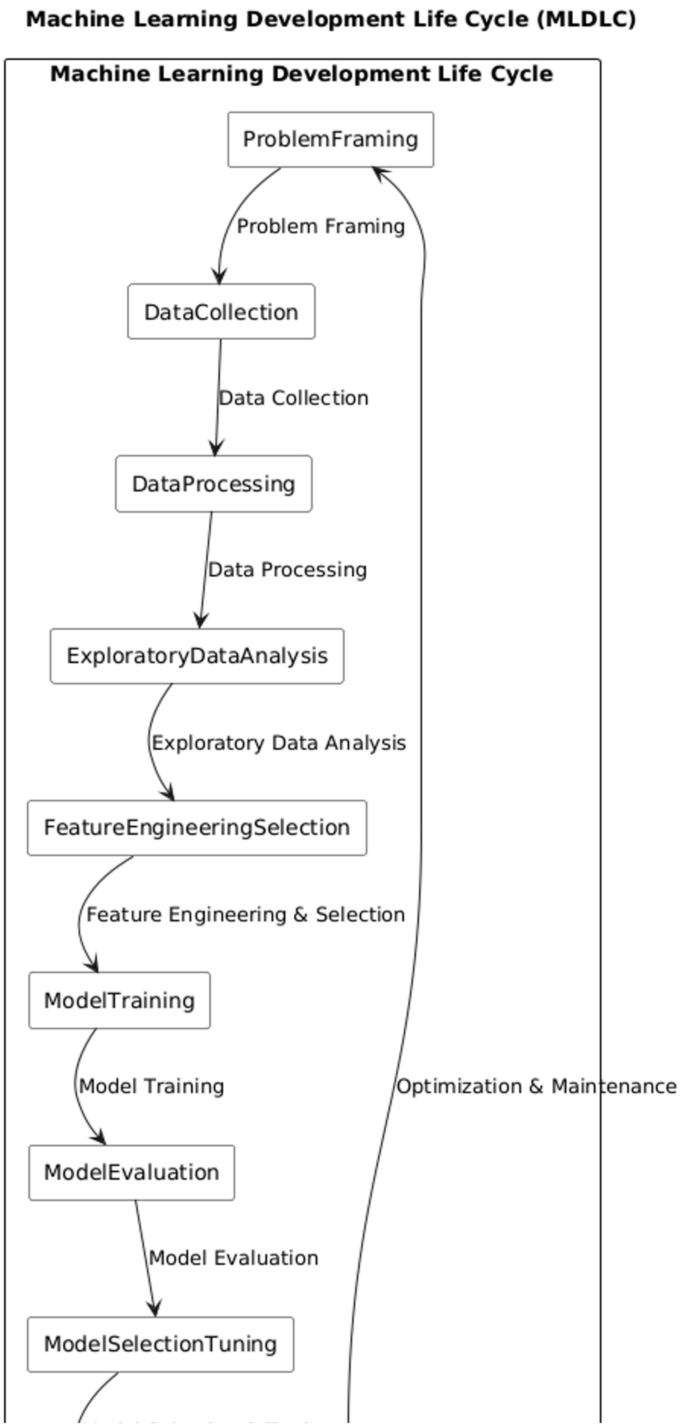
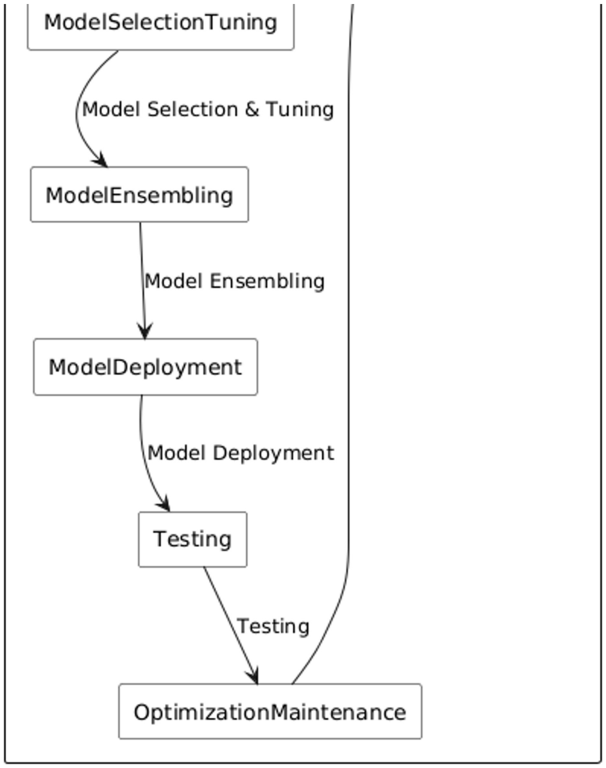
9- Machine Learning Development Life Cycle | MLDLC in Data Science

Machine Learning Development Life Cycle (MLDLC): Detailed Notes

The Machine Learning Development Life Cycle (MLDLC) is a systematic set of steps to design, develop, test, and deploy machine learning models effectively. This structured approach ensures a smooth transition from the initial idea to a fully operational ML-powered product.





1. Problem Framing

- **Purpose**: Clearly define the problem, objectives, and scope.
- **Details**: Decide on the target audience, team size, cost, expected product appearance, and required algorithms (e.g., supervised or unsupervised).
- **Example**: For a recommender system, outline who the users will be, the anticipated use case, and key product objectives.

2. Data Collection

- **Purpose**: Gather data needed for model training.
- **Sources**:
  - **Internal Sources**: Company databases and stored records.
  - **External Sources**: APIs, web scraping, or publicly available datasets.
- **Example**: For a travel recommender system, data can be sourced from travel websites or APIs to gather information on hotels, reviews, and user preferences.

3. Data Processing (Data Cleaning)

- **Purpose**: Ensure data is error-free, standardized, and in a consistent format for ML models.
- **Tasks**:
  - Remove duplicates, handle missing values, and standardize scales.
  - Example: A dataset with both small and large numerical values should be scaled to ensure ML algorithms perform accurately.

4. Exploratory Data Analysis (EDA)

- **Purpose**: Understand data characteristics and relationships.
- **Methods**:
  - Use statistical analysis, visualizations, and correlation analysis to uncover relationships.
- **Example**: Visualize distribution patterns in housing prices or the correlation between features (e.g., area vs. price).

→ Univariate / Biavariate
→ outlier detection
imbalance dataset handling

→ outlier detection
→ imbalance dataset handling

5. Feature Engineering and Selection

i/p column

- **Feature Engineering**: Create new relevant features, such as combining "number of rooms" and "number of bathrooms" to generate "total square footage." → generated feature
- **Feature Selection**: Retain only the most impactful features for the model.
- **Example**: In a housing price prediction model, using square footage instead of separate room counts simplifies and optimizes the dataset.

6. Model Training

- **Purpose**: Train multiple models to identify the best-performing one.
- **Approach**:
  - Run data through different algorithms to compare performances.
  - Example: Try decision trees, logistic regression, and neural networks for a classification problem, observing which provides the highest accuracy.

7. Model Evaluation

- **Purpose**: Measure model performance with evaluation metrics.
- **Metrics**:
  - Classification accuracy, precision, recall, F1-score, etc.
  - Regression metrics like mean squared error (MSE) or R-squared (R^2).
- **Example**: For a classification problem, if the model achieves high precision but low recall, adjustments in model design may be required.

8. Model Selection and Hyperparameter Tuning

- **Purpose**: Choose the best model and optimize parameters.
- **Hyperparameter Tuning**:
  - Adjust parameters like learning rate, max depth, and more to improve model performance.
- **Example**: Use grid search or randomized search for tuning parameters in a neural network to maximize its performance.

9. Model Ensembling (if necessary)

- **Purpose**: Combine multiple models to boost accuracy.
- **Techniques**:
  - Techniques like bagging, boosting, or stacking can enhance performance.
- **Example**: Use ensemble techniques to increase predictive strength in complex tasks like image recognition.

10. Model Deployment

- **Purpose**: Make the model accessible to users.
- **Method**:
  - Deploy as an API or integrate into a software application.
- **Example**: Deploying a model as an API allows external applications to call the model for predictions, facilitating integration with websites, mobile apps, or desktop applications.

11. Testing (Beta Testing)

- **Purpose**: Validate model performance in a real-world setting.
- **Method**:
  - Perform beta testing with a small group to collect feedback.
- **Example**: In a recommendation system, gather feedback from early users to ensure recommendations are accurate and relevant before full deployment.

    - Perform beta testing with a small group to collect feedback.
- **Example**: In a recommendation system, gather feedback from early users to ensure recommendations are accurate and relevant before full deployment.

12. Optimization and Maintenance

- **Purpose**: Regularly improve the model's efficiency and accuracy.
- **Tasks**:
    - Set up data backup, automated retraining, and load balancing.
- **Example**: In fraud detection, continuous retraining may be necessary to stay updated with new types of fraud attempts.

A/B Testing

Quick Revision Notes (Summary of MLDLC Steps)

Step	**Objective**	**Example**
Problem Framing	Define the scope, objectives, and audience.	For recommender systems, specify user demographics and requirements.
Data Collection	Gather necessary data from APIs, web scraping, or internal sources.	For a travel site, scrape hotel data from relevant websites.
Data Processing	Clean and standardize data.	Remove duplicates, handle missing values, and scale numerical features.
Exploratory Data Analysis	Analyze data patterns and relationships.	Visualize relationships between area and price in housing data.
Feature Engineering & Selection	Create new features and keep only essential ones.	Combine "rooms" and "bathrooms" into "square footage" for simpler analysis.
Model Training	Train various models and evaluate each.	Test multiple algorithms on the same dataset to find the best fit.
Model Evaluation	Measure model performance using metrics.	Use precision, recall, and F1-score for classification models.
Model Selection & Tuning	Choose the best model and fine-tune hyperparameters.	Tune learning rate or max depth in a decision tree model.
Model Ensembling	Combine models for enhanced accuracy (if required).	Apply boosting to improve accuracy in image recognition tasks.
Model Deployment	Make model available for real-world use via API or integration.	Deploy recommendation API to be accessible through a website or app.
Testing	Perform beta testing to refine the model.	Run a beta version with limited users for initial feedback.
Optimization & Maintenance	Regularly improve and update the model as needed.	Set up automated retraining for fraud detection models to stay effective against new fraud types.