AI Development Workflow: From Problem Definition to Deployment

Course: AI in Software Engineering
Date: July 2025

Group Members:

1. Gathigi Moses Muiruri -
gathimoses@gmail.com

2. Odongo lsaiah
odongoreagan19@gmail.com

3. Keren Hapuch Ntinyari
kerenhapuch68@gmail.com

4. Jebichii Joyce

jebichiijoyce@gmail.com

5. Palpable Smart:
palpable237@gmail.com

Introduction

The Artificial Intelligence (AI) Development Workflow is a systematic, iterative process that guides the creation of robust and effective AI solutions. It provides a structured framework encompassing every stage from the initial conception of a problem to the final deployment and ongoing maintenance of a model. This workflow ensures that AI projects are aligned with business objectives, are technically sound, and are developed with critical consideration for ethical implications and practical challenges.

This report will demonstrate a comprehensive understanding of this workflow through two parts. **Part 1** will address short-answer questions by defining a hypothetical AI problem and detailing the steps required to solve it. **Part 2** will apply this workflow to a real-world case study: predicting patient readmission risk in a hospital setting, delving into the specific data strategies, ethical considerations, and deployment challenges inherent in the healthcare domain.

Part 1: Short Answer Questions

For this section, we will use a hypothetical but common business problem to illustrate the core stages of the AI development workflow.

1.1 Problem Definition

Hypothetical AI Problem: Predicting Customer Churn for a Subscription-Based Software-as-a-Service (SaaS) Company.

The goal is to proactively identify customers who are at a high risk of canceling their monthly or annual subscription in the near future.

Objectives:

1. **Proactive Customer Retention:** To identify at-risk customers at least 30 days before their subscription renewal date, allowing the customer success team to intervene with targeted support, discounts, or training.

2. **Efficient Resource Allocation:** To focus expensive retention efforts (e.g., personal calls, dedicated support) only on customers with a high predicted probability of churn, rather than applying a one-size-fits-all approach.

3. **Product Improvement Insights:** To understand the key drivers of churn by analyzing which features are most predictive in the model (feature importance). This feedback can inform the product development roadmap.

Stakeholders:

1. **Customer Success & Retention Team:** The primary users of the model's output. They will use the list of at-risk customers to guide their daily activities and engagement strategies.

2. **Product Management Team:** They will use insights from the model (e.g., "customers who don't use Feature X are more likely to churn") to prioritize feature improvements and new developments.

Key Performance Indicator (KPI):

- **Churn Rate Reduction:** The primary business metric to measure the model's success will be the **percentage reduction in the monthly churn rate**. For example, if the baseline churn rate is 4% per month, a successful implementation might aim to reduce it to 3%, a 25% relative reduction.

1.2 Data Collection & Preprocessing

Data Sources:

1. **User Activity/Engagement Data:** This data would be sourced from the application's backend database or event-tracking service (e.g., Mixpanel,

Segment). It includes logs of user actions such as login frequency, features used, time spent on the platform, number of projects created, and support tickets submitted.

2. **Subscription and Billing Data:** This would be sourced from the company's payment processor or CRM (e.g., Stripe, Salesforce). It includes information like subscription plan tier, contract length, customer tenure (how long they have been a customer), and payment history (e.g., late payments).

Potential Bias in Data:

- **Historical Bias:** The model learns from past churn patterns. If the company recently underwent a major change—such as a significant price increase, a major product redesign, or the launch of a highly-requested feature—the historical data may no longer be representative of current churn behavior. The model trained on this old data might incorrectly flag customers or miss new patterns, leading to poor performance. For instance, it might not know that the new feature is a major driver of retention.

Preprocessing Steps:

1. **Handling Missing Data:** User activity data may have missing values. For instance, a field like days_since_last_login could be null for a new user. A robust strategy would be to impute this with a meaningful value, such as zero. For missing numerical features like average_session_duration, we could impute the median value for that customer's subscription tier to avoid skewing the data with outliers.

2. **Feature Engineering:** This is a critical step to create more predictive signals. From raw timestamp data, we can engineer features like:

    - usage_frequency_last_30_days (number of logins/30)
    - feature_adoption_rate (percentage of key features used by the customer)

- time_since_last_support_ticket

These engineered features often have more predictive power than the raw data alone.

3.

4. **Normalization:** Algorithms like Support Vector Machines or Logistic Regression are sensitive to the scale of input features. A feature like customer_tenure (in days, e.g., 1-1000) would numerically dominate a feature like support_tickets_last_month (e.g., 0-5). We would apply a scaling technique like **Min-Max Scaling** to transform all numerical features to a common range (e.g., 0 to 1), ensuring that all features contribute equally to the model's learning process.

1.3 Model Development

Model Choice and Justification:

- **Model: Gradient Boosting Machine (e.g., XGBoost, LightGBM).**

- **Justification:** Gradient Boosting models are an excellent choice for this type of tabular data problem for several reasons:

1. **High Performance:** They are consistently among the top-performing models for classification tasks on structured/tabular data.

2. **Handles Mixed Data Types:** They can effectively handle a mix of numerical and categorical features without extensive pre-processing.

3. **Feature Importance:** These models provide built-in feature importance scores, which directly addresses our objective of understanding the *drivers* of churn. This allows the product team to see exactly which user behaviors or subscription details are most correlated with a customer leaving.

Data Splitting:

The dataset would be split into three distinct sets to ensure robust model development and unbiased evaluation:

1. **Training Set (70%):** The largest portion of the data, used to train the model and allow it to learn the patterns and relationships between the input features and the churn outcome.

2. **Validation Set (15%):** Used to tune the model's hyperparameters and for model selection. The model does not learn from this data; instead, we evaluate the performance of different hyperparameter settings on this set to find the optimal configuration and prevent overfitting to the training data.

3. **Test Set (15%):** This set is held back and used only *once* at the very end of the development process to provide a final, unbiased estimate of the model's performance on unseen data. This simulates how the model would perform in the real world.

Hyperparameter Tuning:

1. **n_estimators (Number of Trees):** This hyperparameter controls the number of sequential trees built in the model. A low number can lead to underfitting (the model is too simple), while a very high number can lead to overfitting (the model memorizes the training data) and increases computation time. We would tune this to find the sweet spot where performance on the validation set is maximized.

2. **learning_rate:** This controls the step size at which the model corrects the errors of the previous trees. A lower learning rate makes the model more robust to overfitting but requires more trees (n_estimators) to achieve good performance. Tuning this in conjunction with n_estimators is crucial for finding an optimal balance between accuracy and training speed.

1.4 Evaluation & Deployment

Evaluation Metrics:

1. **Precision:** This metric answers the question: "Of all the customers the model predicted would churn, what percentage actually churned?" Precision = True Positives / (True Positives + False Positives). High precision is important for **efficient resource allocation**. It ensures that the customer success team is not wasting time and resources on customers who were never going to leave.

2. **Recall (Sensitivity):** This metric answers the question: "Of all the customers who actually churned, what percentage did the model correctly identify?" Recall = True Positives / (True Positives + False Negatives). High recall is critical for **effective retention**. The primary goal is to catch as many at-risk customers as possible, so we want to minimize the number of "misses" (False Negatives). There is often a trade-off between precision and recall that must be balanced based on business priorities.

Concept Drift and Monitoring:

- **What is Concept Drift?** Concept drift is the phenomenon where the statistical properties of the target variable (in this case, churn) change over time. This means the relationships between input features and the outcome that the model learned are no longer valid. For example, a new competitor entering the market could introduce new reasons for churn that the model has never seen before, making its predictions inaccurate.

- **Monitoring:** To monitor for concept drift post-deployment, we would implement two strategies:

    1. **Performance Monitoring:** Continuously track the model's precision and recall on new data. A sustained drop in these metrics is a strong indicator of drift.

    2. **Data Distribution Monitoring:** Track the statistical distributions of key input features (e.g., average login frequency). If the distribution of a key feature changes significantly, it's a warning that the incoming data is different from the training data, and the model may need to be retrained.

Technical Challenge during Deployment:

- **Latency for Real-Time Prediction:** A significant challenge is deciding between batch and real-time deployment. If the stakeholder (e.g., a sales agent on a call) needs an instantaneous churn score for a customer, the model must be deployed as a **low-latency API endpoint**. This requires more complex infrastructure (e.g., a containerized service on Kubernetes, a serverless function) capable of returning a prediction in milliseconds. In contrast, a batch system that simply generates a daily report of at-risk customers is simpler to build but less flexible. Ensuring the real-time system is scalable, available, and fast is a major technical hurdle.

Part 2: Case Study Application - Predicting Patient Readmission Risk

Scenario: A hospital wants an AI system to predict patient readmission risk within 30 days of discharge.

2.1 Problem Scope

Problem Definition:
To develop and deploy a reliable, interpretable, and fair machine learning model that predicts the 30-day readmission risk for patients at the time of their discharge from the hospital. The output should be a risk score or probability that can be integrated into clinical workflows.

Objectives:

1. **Improve Patient Outcomes:** Identify high-risk patients to enable targeted post-discharge interventions, such as follow-up calls from nurses, home health visits, or medication reconciliation, thereby reducing preventable readmissions and improving patient health.

2. **Reduce Healthcare Costs:** Minimize financial penalties imposed on hospitals for high readmission rates and reduce the overall cost burden on the healthcare system associated with unplanned hospital stays.

3. **Optimize Clinical Resource Allocation:** Ensure that limited clinical resources, such as case managers and discharge planners, are directed towards the patients who are most in need of post-discharge support.

Stakeholders:

- **Clinicians (Doctors, Nurses):** They will use the risk score as a decision-support tool to inform their discharge summaries and care plans.
- **Case Managers/Discharge Planners:** The primary users of the system. They will use the risk score to prioritize patients for intensive post-discharge planning and follow-up.
- **Hospital Administrators:** They will monitor the model's impact on the hospital's overall readmission rates, financial performance, and quality-of-care metrics.
- **Patients and Their Families:** The ultimate beneficiaries, who receive better-coordinated care and have a lower chance of experiencing a disruptive and costly hospital readmission.

2.2 Data Strategy

Proposed Data Sources:

1. **Electronic Health Records (EHR):** The primary source, containing rich clinical information:
    - **Clinical Data:** Diagnoses (ICD-10 codes), procedures performed, vital signs, lab results (e.g., A1c levels, creatinine).
    - **Medication Data:** Prescribed medications at discharge.
    - **Hospitalization History:** Length of stay, admission type (emergency vs. elective), number of prior admissions in the last 12 months.

2.
3. **Demographic Data:** Patient's age, gender, and insurance type.

4. **Discharge Information:** Discharge disposition (e.g., discharged to home, skilled nursing facility, home with health services).

Ethical Concerns:

1. **Patient Privacy and HIPAA:** EHR data is highly sensitive and is protected under regulations like the Health Insurance Portability and Accountability Act (HIPAA) in the U.S. There is a significant ethical and legal obligation to protect this data from unauthorized access or breaches. All data must be de-identified before model training, and access to the final system must be strictly controlled and audited.

2. **Algorithmic Bias and Fairness:** The model could perpetuate or even amplify existing health inequities. For example, if historical data shows that patients from a certain zip code (a proxy for socioeconomic status) have higher readmission rates due to systemic factors like lack of access to primary care, the model might learn to flag all patients from that area as high-risk. This could lead to stigmatization and unfairly allocate resources, without addressing the root cause. It is crucial to audit the model for fairness across different demographic groups.

Preprocessing Pipeline:

1. **De-identification and Integration:** Combine data from the sources above. Remove all Protected Health Information (PHI), such as names, addresses, and social security numbers, replacing them with a unique, non-identifiable patient ID.

2. **Feature Engineering:**

    - **Comorbidity Index:** Create a feature like the Charlson Comorbidity Index by counting the number of chronic conditions a patient has based on their ICD-10 codes.

- **Prior Utilization:** Engineer features such as number_of_emergency_visits_last_6_months and total_hospital_days_last_year.

- **Categorical Feature Consolidation:** Group thousands of specific ICD-10 codes into broader, clinically relevant categories (e.g., "Cardiovascular Disease," "Respiratory Disease," "Diabetes-related").

3.
4. **Handling Missing Data:** Use clinically-informed imputation. For a missing lab value, impute the median value for patients in a similar demographic and diagnostic group. For a feature like number_of_prior_admissions, a missing value likely means zero and should be imputed as such.

5. **Encoding and Scaling:**

   - Apply **One-Hot Encoding** to categorical features like discharge_disposition and insurance_type.

   - Apply **Standard Scaling** (Z-score normalization) to all numerical features to prepare them for the model.

2.3 Model Development

Model Selection and Justification:

- **Model: Logistic Regression.**

- **Justification:** While more complex models like XGBoost might achieve slightly higher accuracy, **interpretability is paramount in a clinical setting.** A Logistic Regression model is highly interpretable. It produces coefficients for each feature, which can be translated into odds ratios. This allows a clinician to understand *why* the model assigned a certain risk score. For example, the model can explicitly show that "an increase of 1 in the number of prior admissions increases the odds of readmission by 25%." This transparency builds trust with clinicians and makes the model's output

actionable and defensible, which is a critical requirement for a decision-support tool in healthcare.

Confusion Matrix and Metrics (Hypothetical Data):

Assume we evaluate our model on a test set of 1,000 patients. The results are:

	Predicted: No Readmission		**Predicted: Readmission**		**Total Actual**
Actual: No Readmission	TN	= 850	FP 50	=	900
Actual: Readmission	FN	= 20	TP 80	=	100
Total Predicted	870		130		1000

Calculation of Precision and Recall:

- **Precision:** TP / (TP + FP) = 80 / (80 + 50) = 80 / 130 ≈ 61.5%
  - **Relevance:** When our model flags a patient as high-risk, it is correct 61.5% of the time. This metric is important to ensure that the limited resources of case managers are not being wasted on a large number of false alarms (False Positives).
-
- **Recall:** TP / (TP + FN) = 80 / (80 + 20) = 80 / 100 = 80%
  - **Relevance:** Of all the patients who were actually readmitted, our model successfully identified 80% of them. In a clinical context,

recall is often the more critical metric. Missing a high-risk patient (a False Negative) could lead to a preventable adverse health event. It is generally better to have a few false alarms than to miss a patient who genuinely needs help.

2.4 Deployment

Integration Steps:

1. **Develop a Secure, HIPAA-Compliant API:** The trained Logistic Regression model will be packaged and deployed as a secure web service (API) on a HIPAA-compliant infrastructure (e.g., AWS Healthcare Cloud, on-premise secure server).

2. **EHR System Integration:** The hospital's IT department will work to integrate this API into the existing EHR system. The API will be called with a patient's de-identified data at the time of discharge planning.

3. **Clinical Workflow Integration:** The returned risk score will be displayed directly within the patient's chart in the EHR, ideally with a visual cue (e.g., a red/yellow/green flag) to quickly alert the discharge planner.

4. **Training and Protocol Development:** A protocol will be developed defining the actions to be taken for patients flagged as high-risk (e.g., mandatory consultation with a case manager). All relevant staff (nurses, doctors, case managers) will be trained on how to interpret the score and follow the new protocol.

Ensuring HIPAA Compliance:

1. **Data De-identification:** The model must be trained and must make predictions using de-identified data. The API should only accept a non-identifiable patient ID and the required clinical features.

2. **Role-Based Access Control (RBAC):** Only authorized clinical personnel with a legitimate need-to-know should be able to view the risk scores within the EHR.

Access must be tied to their login credentials.

3. **Secure Infrastructure:** The model and API must be hosted on servers that meet HIPAA's technical safeguards, including data encryption at rest and in transit, and robust network security.

4. **Audit Trails:** The system must log every time a risk score is requested and viewed, including which user accessed it and when. This creates an audit trail for accountability.

2.5 Optimization

Method to Address Overfitting:

- **Method: L2 Regularization (Ridge Regression).**

- **Proposal:** Overfitting occurs when a model learns the noise in the training data rather than the true underlying signal, causing it to perform poorly on new, unseen data. For a Logistic Regression model, this can happen when coefficients for certain features become excessively large. **L2 Regularization** adds a penalty term to the model's cost function that is proportional to the square of the magnitude of the coefficients. This penalizes large coefficients, effectively shrinking them towards zero. By applying L2 Regularization, we force the model to be more conservative and prevent it from placing too much importance on any single feature. This results in a simpler, more robust model that is more likely to generalize well to the broader patient population, improving its real-world performance and reliability.

Part 3: Critical Thinking

1. Ethics &Bias

   Bias refers to an subjective opinion, preference,or inclination that influences individual.

   Impact of bias can be negative leading to.

- unfair outcomes, discrimination, unequal opportunity
- Erosion of trust, undermining faith in institution.
- Perpetuation of inequality: reinforcing existing social and economic disparities.

Ethics,it's the philosophical study of moral phenomena .

Key aspects of ethics

- moral principles and values, honesty
- Decision making frameworks , Tools and theories for analyzing moral dilemmas.

☐ How might biased training data affect patient outcomes in case study.

- Misallocation of Resources: over attention to false positive and under attention to false negatives, meaning limited resources aren't directed where there most needed
- Worsening Health disparities; Reinforcing and exacerbating existing inequalities in healthcare access and outcomes for already marginalized groups.
- Suboptimal patient care: either through unnecessary interventions or more critical by failing to provide crucial support to those who truly need it.
- Erosion of Trust: patients and health care providers may lose faith in AI systems if they consistently produced inaccurate results

Suggest 1 strategy to mitigate this bias

Model -centric strategies (During AI model Development).

1)Feature selection and Engineering with Fairness in mind.

- Avoid proxies for protected attributes, carefully review features that might act as proxies for protected attributes e.g zip codes, specific clinics

2) Fairness Awareness machine learning Algorithms :

☐ in processing techniques : use algorithms designed to incorporate fairness constraints during training examples including

- Fairness Regularization: Adding a penalty term to the model objectives function that discourages disparate outcome across groups.
- Adversarial Debiasing: Training a de-biasing components to remove information about models representative, while still allowing the main model to predict readmission.

2. Trade offs.

- ☐ Discuss the trade-off between model interpretability and accuracy in healthcare
  - Accuracy, means how well the AI model performs it's intended task .in healthcare this could mean correctly diagnosing a disease, accurately predicting a patient risks of readmission or previously identifying cancerous cells in an image. Accuracy is measured by metrics like precision, recall F1 core.
  - Interpretability: This refers to degree to which a human can understand the reasoning behind a model decision or predictions. it allows clinicals to answer questions like:
  - why did the model predict this patient has a high risk of sepsis ?
  - under what conditions would the models prediction change ?.
- ☐ Trade off in Healthcare:
  - Complex models (often black box) tend to achieve higher accuracy: modern machine learning techniques , particularly deep learning , are excellent at identifying subtle , non linear patterns in vast datasets that simpler models or even human human experts might miss.This can lead to superior diagnostic , accuracy , more precise risk predictions. Examples, Image Recognition: Deep learning models can achieve near human(or even super human) accuracy in detecting subtle signs of disease from medical images.e.g identifying diabetic retinopathy from retinal scans ,detecting pneumonia on X-ray.
  - Complex Disease prediction: for disease with many interacting factors , complex models might better capture the intricate relationships.

Simpler models (interpretable)often have lower accuracy, while linear models or simple decisions trees are easy to understand, they might oversimplify complex biologicial or clinical pneumonia.

Why interpretability is crucial in Healrhcare:

- Trust and adoption : Clinicians are ethically and legally responsible for patients outcomes.They need to understand why an AI model made a particular recommendations before they can trust and adopt it into their practice".A black box" recommend.
- Clinical Validation and safety : if model makes an incorrect prediction , an interpretable model allows clinicians to understand why it erred.
- informed consent: patients have a right to understand why certain medical decisions are being made about their care.
- legal and Regulatory Requirements: As Al system become more prevalent regulatory bodies e.g FDA in the US , EMA in Europe) and legal frameworks and increasing demanding transparency
- New clinical insights and Knowledge Discovery: interpretable models can sometimes reveal new, previously unrecognized patterns or relationships in clinical data.
- Personalized medicine: For truly personalized care , clinicians need to understand how various patients' specific factors contribute to a recommendation.

Strategies to address to the trade-off:

- Hybrid Approaches:

  Post hoc explainability (XAI) : use techniques that explains the predictions of complex "black box " model after they have been trained.Examples

  SHAP (shapley Additive explanations) Assigns an important value to each feature for particular prediction.

  LIME(local interpretable model agnostic explanations) explains individuals prediction of any classifier by approximating it locally with an interpretable model

  Attending mechanisms, highlights which part of an image

- Constraints based learning: ensuring that resulting models is both accurate and interpretable.
- Context specific Decisionsv: The acceptable level of interpretability often depends on the clinical context.
- Human in the loop : ensure that Al models are designed as decisions support tools .

If the hospital has limited computational resources, how might this impact model choice?

1. Prefence for simpler, less Resources intensive models:

- Impact : hospital can deploy these models on existing infrastructure without major upgrades.Training can be done frequently to adapt to new data.
- Gradient boosting Machines; these are powerful ensembles methods that often achieve very high accuracy and are commonly used in tabular data problems like readmission prediction. ok impact : a strong contender when accuracy is critical but GPU resources are scarce.

2) Avoidance of Deep Learning Models (Neural Network) :

- High Computational cost : Deep learning models, especially those with many layers and parameters e.g complex Recurrent Neural Network for time series EHR.
- Long Training Times: Training derp learning models can take hours , day's or even weeks on powerful GPUs.
- Large memory Footprint: derp learning models can also consume substantial amounts of RAM, both for model parameters and for processing large batches of data during training. impact hospitals with limited resources would likely rule out deep learning unless they can secure cloud based GPU access.
- Trade off : missing out on potential gains in predictive accuracy that derp learning might offer , especially if the underlying patterns in the data are highly non linear or if unstructured data.

3) Implications for Deployment and inference:

- Real time prediction constraints: even after training , complex models can have higher inference costs (the resources needed to make a single prediction) impact ; Hospital would prioritize models with fast inference times , even if it means a slight hit to accuracy.

4) Cloud vs on premise solutions:

- Cloud computing as workaround : Hospitals with limited on premise resources might consider cloud based AI services e.g Google cloud AI platforms. impact : shift capital expenditure to operational expenditure can provide Access to powerful GPUs when needed for training.

5) Data size and Feature Engineering:

- impact on Features engineering: with limited computational resources, hospitals might be less able to experiment with extensive features engineering (creating new features from raw data) which can be computationally intensive itself.
- Smaller Datasets: if training on extremely large datasets (billions of data points) ,even simpler models can become resource - intensive . Hospitals might need to work with smaller , more manageable subsets of data, which could affect model organizability.

6) Fewer complex deep learning

7) A strong evaluation of the cost benefit of cloud computing for training versus the limitations of on premise solutions.

Part 4: Reflection & Workflow Diagram

1.Reflection:

What was the most challenging part of the workflow ? why ?

- Communication Breakdown: lack of clear instructions, incomplete information can lead to delays or mistakes. Misunderstanding can occur when tasks are passed from one person to another.
- inconsistent process: Manual workaround or unclear systems can cause bottlenecks . Without standardized procedures each person might complete tasks differently.
- Lack of Accountability: When responsibility are not clearly defined, tasks may fall through the cracks. All might assume someone else is doing the tasks.
- Tool integration issues: workflows often span multiple tools or platforms . poor integration might interrupt the flow.
- Delays in approval of Feedback: if deadlines aren't enforced or reminders aren't automated , progress can halt.
- Difficulty in understanding intent : it's hard to determine what kind of information the user is truly seeking.
- inability to provide precise assistance: primary goal is to provide accurate and helpful Ambiguity directly hinders my ability to be precise and deliver the most useful answer possible.
- Risk of irrelevant information: without a clear understanding of user's intent , my response might contain information that isn't relevant to their actual need, making the answer less helpful or frustrating.

How would you improve your approach with more time/; resources.

- ☐ Deepen planning and mapping: spend more time clearly mapping the entire workflow from beginning to end.Benefit save time and reduce human error.
- ☐ Automate Repetitive Tasks: use tools like Notions to eliminate manual steps.
- ☐ Clarify on Roles and Responsibilities: Assign clear roles of ownership for each tasks . boost accountability and minimize confusion.
- ☐ Build in Feedback Loops : schedule regular reviews or check ins to evaluate progress.
- ☐ Provide Training and Resources: Equip team members with better onboarding or documentation.
- ☐ More dynamic and Adaptive learning
    - Raid skill Acquisition, l could learn new skills or adapt to new types of tasks much faster , requiring less specific training data for novel challenges.
    - self correction mechanism , ability to identify and self correct errors in my own reasoning.

2. Diagram

sketch a flow chart of AI Development workflow, labelling all stages

Flowchart: AI Development Workflow

Start

↓

1.)Problem Definition

↓

2.)Data Collection

↓

3.)Data Preprocessing

↓

4.)Exploratory Data Analysis (EDA)

↓

5.)Feature Engineering

↓

6.)Model Selection

↓

7.)Model Training

↓

8.)Model Evaluation

↓

9.) Hyperparameter Tuning

↓

10.) Model Validation

↓

11.)Model Deployment

↓

12.)Monitoring & Maintenance

↓

13.)Feedback & Continuous Improvement

↓

14.)End

Explanation of Each Stage:

1. Problem Definition – Understand the objective, scope, and expected outcomes.

2. Data Collection – Gather relevant raw data from various sources.

3. Data Preprocessing – Clean, normalize, and transform data into usable format.

4. Exploratory Data Analysis (EDA) – Identify patterns, anomalies, and relationships.

5. Feature Engineering – Create, select, and refine variables to improve model performance.

6. Model Selection – Choose the appropriate AI/ML algorithm(s).

7. Model Training – Train the model on prepared data.

8. Model Evaluation – Test model performance using metrics (accuracy, F1-score, etc.).

9. Hyperparameter Tuning – Adjust settings to optimize model results.

10. Model Validation – Ensure generalization with cross-validation or test sets.

11. Model Deployment – Integrate model into production environment.

12. Monitoring & Maintenance – Track performance over time and fix issues.

13. Feedback & Continuous Improvement – Use real-world results to improve the model iteratively.