🔷 1. Problem Framing

- Define business objective
- Understand usage & impact
- Identify problem type (regression, classification, clustering)
- ✓ NEW: Decide learning approach (supervised, unsupervised, self-supervised, reinforcement learning)
- Define performance metrics (e.g., RMSE, F1, Accuracy)
- ✓ NEW: Capture real-world constraints (latency, hardware, interpretability)
- ✓ NEW: Document assumptions and verify if possible

How to Identify the Problem Type:

- **Regression**: Predicting continuous numeric values (e.g., price, temperature)
- **Classification**: Predicting categories/labels (e.g., spam or not, cancer or not)
- **Clustering**: Grouping similar data points without labels (e.g., customer segmentation)

Choosing a Learning Type:

- **Supervised Learning**: Labeled data (e.g., classification, regression)
- **Unsupervised Learning**: No labels (e.g., clustering, dimensionality reduction)
- **Self-Supervised Learning**: Create pseudo-labels from data itself (e.g., contrastive learning, embeddings)
- **Reinforcement Learning**: Learn by interacting with an environment and receiving feedback
- **Deep Learning**: Apply when working with high-dimensional data (images, text, speech) using neural networks (e.g., CNN, RNN, Transformer)

🔷 2. Data Collection

- List required data and sources
- Acquire & store data (DBs, APIs, CSVs, etc.)
- ✓ NEW: Track data versioning (e.g., DVC, Delta Lake)
- ✓ NEW: Consider data labeling workflows for supervised learning

- Ensure sensitive data is masked/anonymized
- Assess volume, variety, legal permissions
- Split off untouched test set early (save separately)

◇ 3. Exploratory Data Analysis (EDA)

- Understand column types, value distributions
- Identify missing values, outliers, invalid values
- Use histograms, boxplots, scatter plots, heatmaps
- ✓ NEW: Check for data leakage
- ✓ NEW: Explore target variable (imbalance, skewness, etc.)
- Document EDA findings (Notion, Markdown, PDF, etc.)

◇ 4. Data Preparation

- Create clean working copy
- Handle missing values (mean/median/mode/drop)
- Impute categorical using mode or "Unknown"
- Remove zero-variance columns
- Fix inconsistent values (e.g., case, format)
- Encode categorical columns (Label, One-Hot, Frequency)
- Convert boolean to integers if needed
- Scale numeric features (StandardScaler / MinMaxScaler)
- ✓ NEW: Use sklearn.Pipeline for reusable preprocessing
- ✓ NEW: Modularize all transformation functions for reproducibility

☑ Baseline Modeling Prep

- If using GridSearchCV or cross_val_score, no need for manual validation set

Summary:

- **Baseline model only**: train_test_split is enough
- **Manual hyperparameter tuning**: use a validation set
- **Using GridSearchCV/AutoML**: no explicit val set needed
- **Deep Learning**: split into train/val/test

◇ 5. Baseline Modeling

- Select a simple base model (e.g., Linear Regression, Decision Tree)
- Split train/test (e.g., 80/20)
- ✔️ NEW: Use stratified split for classification
- ✔️ NEW: Set random seed for reproducibility
- Evaluate with default hyperparameters
- Save baseline metrics (RMSE, MAE, accuracy, etc.)
- Define your target and features

🔜 What Comes Next:

1. 🔍 **Analyze Performance**
   a. Check metrics: R^2, RMSE, MAE
   b. Examine underfitting/overfitting
   c. Compare training vs test accuracy
2. 🧪 **Try Other Models**
   a. Try DecisionTreeRegressor, RandomForestRegressor, XGBRegressor, etc.
   b. Compare 3–5 models and record CV scores
3. ⚙️ **Hyperparameter Tuning**
   a. Use GridSearchCV or RandomizedSearchCV
   b. Tune parameters like depth, estimators, etc.
4. 📈 **Pick Best Model**
   a. Based on CV performance, simplicity, and stability
5. 🧠 **Model Explainability (Optional)**
   a. Use SHAP, LIME, .coef_, or feature_importances_
6. ✅ **Final Predictions**
   a. Retrain best model on full train set
   b. Predict on test or new unseen data

◇ 6. Model Tuning & Ensembling

- Use GridSearchCV or RandomizedSearchCV
- Include preprocessing in pipeline
- Try RandomForest, XGBoost, SVM, etc.

- ✓ NEW: Use AutoML tools (AutoSklearn, H2O, TPOT, etc.)
- ✓ NEW: Ensemble top models (Voting, Stacking, Blending)
- Track training logs and hyperparameter search space

◇ 7. Final Model & Prediction

- Train final model on full training data
- Predict on holdout/test set
- Save predictions to file
- ✓ NEW: Validate model fairness (gender, age, etc.)
- ✓ NEW: Use SHAP / LIME / ELI5 for explainability
- Save model as .pkl or .joblib

◇ 8. Presentation

- Summarize key findings
- Visualize metrics, feature importance, and error types
- ✓ NEW: Create dashboards with Streamlit / Dash
- Prepare a slide deck or business-friendly document
- Communicate business impact clearly ("So what?")

◇ 9. Deployment & Monitoring

- Deploy using Flask, FastAPI, or cloud (AWS, GCP, Azure)
- ✓ NEW: Use CI/CD tools (MLflow, GitHub Actions, Vertex AI)
- ✓ NEW: Set up monitoring (drift, accuracy, input quality)
- ✓ NEW: Include rollback strategy
- Log requests and predictions
- Document API contract and endpoints

◇ 10. Interactive Frontend / UV UI Integration 💬

- Build lightweight interface using Streamlit, Dash, Gradio, or UV (Universal Viewer)
- Connect frontend input fields (dropdowns, sliders) to model input schema

- Display model output with prediction, confidence, and explanation
- ✓ Optional: Add SHAP or LIME visualizations
- ✓ Optional: Style and format for domain-specific output (₹, %, etc.)
- ✓ Optional: Log user inputs/outputs for future improvement

📅 Example:

- User selects: Make=Honda, Year=2018, Fuel=Petrol
- Model predicts: ₹5.2 Lakhs
- Output includes SHAP: "Year -₹0.6L, Fuel +₹0.3L"

◇ 11. Continuous Learning (Optional)

- Monitor model degradation over time
- Schedule regular retraining (weekly, monthly, on-demand)
- ✓ Maintain version history of data and models
- ✓ Build feedback loops for user-flagged errors
- Auto-label and retrain with new incoming data

◇ Generative AI & MCP (Model Context Protocol)

- Integrate LLM-based components to generate explanations, summaries, or recommendations
- Use MCP to pass structured metadata between model stages for traceability
- Examples: Explain prediction rationale with GPT, generate auto-reports, or context-aware input transformations