DA 204o: Data Science in Practice

Course Project Proposal

Integrated Machine Learning Framework for Soil Moisture Prediction and Classification Across India

Patel Parth Pravinkumar, IISc, parthpatel@iisc.ac.in
Payal Dey, IISc, payaldey@iisc.ac.in
Inderjit Singh Chauhan, IISc, inderjits@iisc.ac.in

Image source: Internet

Course Project

- Compulsory!
- Marks: 20%
- Team size: 3-4
- Duration: ~6 weeks (Oct 15th to Nov 30th)
- Initial proposal: Oct 11th
  - Team formation (choose among yourself)
  - Select project topic/domain and/or datasets
- Final project proposal: Oct 18th
  - Detailed information: Problem definition, dataset(s), proposed methodology, and implementation plan.
  - Submission of slides (use the following slides)
- Checkpoints
  - First: Completion of data preparation and EDA (5%)
  - Second: Completion of model development and validation (5%)
  - Final: Final report, project presentation and demonstration (5%)
  - Peer feedback: 5%

Problem Definition

- Background of the problem
  - India’s agriculture heavily depends on timely and sufficient soil moisture. However, farmers and planners often lack accurate, district-level insights on soil moisture variations across seasons and years.
  - This leads to inefficient irrigation, crop loss, and poor drought management.
- Why is it important?
  - Ensures optimal irrigation, saving water and costs.
  - Reduces crop losses due to under- or over-watering.
  - Helps policymakers and planners monitor drought risk and plan interventions.
  - Supports sustainable agriculture and resource management.
- Objectives of the project
  - Predict average soil moisture levels for districts across India (Regression).
  - Classify districts into Low, Medium, or High soil moisture zones (Classification).
  - Provide actionable insights for irrigation planning and drought management.
  - Build a predictive model that can be updated with future data for ongoing monitoring.
- How can Data Science solve the problem?
  - Use historical soil moisture data to identify patterns and trends.
  - Apply machine learning models (Regression, Classification, Neural Networks) to predict future soil moisture levels and categories.
  - Use derived features (month, lag values, rolling averages) to improve accuracy.
  - Present predictions in dashboards for decision support, allowing data-driven planning for farmers and policymakers.

Data Collection and Preparation

- Data source(s) (where it's from, how it was collected)
  - Soil moisture datasets from Indian government surveys (2018 & 2020).
  - Collected from field measurements across all districts in India.
- Description of the data (features, size, format)
  - Features: Date, State, District, Avg Soil Moisture Level, Avg Soil Moisture Volume, Aggregate %, Volume %.
  - Size: ~2 lakh rows.
  - Format: CSV / tabular numeric + categorical data.
- Any preprocessing steps required
  - Handle missing values.
  - Convert dates to useful features (month, day of year).
  - Create lag features and rolling averages.
  - Encode categorical features (State, District) if required.
  - Normalize / scale numeric features for model training

Proposed Methodology

- Overview of methods or models you plan to use
  - **Regression:** Linear Regression, Random Forest, Gradient Boosted Models (XGBoost, CatBoost, LightGBM) to predict average soil moisture.
  - **Classification:** Random Forest or Gradient Boosted Classifier to categorize districts into Low / Medium / High soil moisture.
  - **Neural Networks:** MLP (Multilayer Perceptron), optionally LSTM for temporal modeling.
- Justification for choosing these methods, if any
  - **Tree-based models** handle non-linear relationships and missing values well.
  - **Neural Networks** capture complex patterns and temporal dependencies.
  - **Baseline models** (e.g., historical averages) provide comparison for evaluation.
- Tools/Technologies (e.g., Python, libraries)
  - **Language:** Python
  - **Libraries:** pandas, numpy, scikit-learn, XGBoost, CatBoost, LightGBM, TensorFlow / PyTorch
  - **Visualization & Deployment:** Matplotlib, Seaborn, Plotly Dash or Streamlit

Implementation Plan

- Project phases (data collection, model building, testing)

  - **Data Collection & Preprocessing (Week 1–2)**
    - Gather soil moisture dataset (2018 & 2020)
    - Handle missing values, create derived features, encode categorical data.
  - **Exploratory Data Analysis & Feature Engineering (Week 2–3)**
    - Visualize trends, correlations, and anomalies
    - Create lag features, rolling averages, moisture categories
  - **Model Building (Week 3–4)**
    - Regression models: Linear Regression, Random Forest, XGBoost, CatBoost
    - Classification models: Random Forest / Gradient Boosted Classifier
    - Neural Networks: MLP / optional LSTM
  - **Model Testing & Evaluation (Week 4–5)**
    - Use temporal split (train 2018, test 2020)
    - Evaluate using MAE, RMSE, R^2 (regression), Accuracy, F1-score (classification)
  - **Insights, Dashboard & Reporting (Week 5–6)**
    - Prepare visualizations and dashboards (Plotly Dash / Streamlit)
    - Summarize key findings and recommendations

Implementation Plan

- Timeline and milestones
  - **Week 1–2:** Data collected and preprocessed.
  - **Week 2–3:** EDA and feature engineering completed.
  - **Week 3–4:** Models built and trained.
  - **Week 4–5:** Models tested, evaluated, and tuned.
  - **Week 5–6:** Dashboard ready and final report submitted.

- Resources required
  - **Hardware:** Laptop with sufficient RAM (16GB+) and GPU (optional for NN)
  - **Software:** Python, Jupyter Notebook / Colab, relevant ML libraries
  - **Data:** Soil moisture dataset (2018 & 2020)
  - **Team:** Parth(preprocessing, model building), Payal (EDA, visualization), Inderjit (classification, reporting, dashboard support)

Challenges and Risks

- Potential risks or challenges
  - **Missing or inconsistent data** – gaps in soil moisture measurements across districts or years.
  - **Overfitting / poor model generalization** – models may perform well on training data but poorly on unseen data.
  - **Feature relevance** – some features may not contribute meaningfully to predictions.
  - **Time constraints** – limited 6-week duration for data processing, modeling, and reporting.

- How you plan to mitigate them
  - **Missing data:** Impute missing values, use robust models (tree-based) that handle gaps.
  - **Overfitting:** Apply cross-validation, regularization, and test on 2020 dataset (temporal split).
  - **Feature relevance:** Perform feature importance analysis, remove irrelevant/redundant features.
  - **Time management:** Follow detailed project timeline, prioritize critical tasks first.

Expected Outcome

- What do you expect to achieve?
  - Accurate prediction of district-wise soil moisture levels (regression).
  - Reliable classification of districts into Low, Medium, High soil moisture zones.
  - Visual insights and dashboards to aid irrigation planning and drought management.
  - Identification of key factors influencing soil moisture across regions.

- How will you measure success?
  - **Regression:** Low MAE / RMSE, high R^2 compared to baseline (historical averages).
  - **Classification:** High Accuracy and F1-score compared to baseline (majority class).
  - **Practical impact:** Predictions are actionable and useful for farmers, planners, and policymakers.

Role and Responsibilities

- Parth:
  - Data Cleaning, Feature Engineering, Regression Modeling, Neural Network, Hyperparameter Tuning, Cross-Validation, Model Evaluation, Interpretation, Lead Analysis
- Payal:
  - EDA, Visualization, Trend Analysis, Heatmaps, Choropleth Maps, Rolling Features, Data Interpretation
- Inderjit:
  - Classification Modeling, Ensemble Learning, Hyperparameter Tuning, Model Evaluation, Performance Metrics, Data-Driven Insights

Machine Learning Canvas

PREDICTIONS	OBJECTIVES	DATA

End-user

Who will use the predictive system / who will be affected by it?

Farmers, irrigation planners, agricultural policymakers, NGOs focused on drought management.

Value proposition

What are we trying to do for the system's users? (e.g. spend less time on X, increase Y...)

Help users optimize irrigation scheduling, reduce crop losses, and anticipate droughts by providing predictive soil moisture insights.

Data sources

Where do/can we get data from? (internal database, 3rd party API, etc.)

Government soil moisture surveys / databases.

Problem

Question to predict answers to (on behalf of user)

Predict and categorize soil moisture to guide irrigation a

Input (i.e. question "parameter")

Date, State, District, historical soil moisture levels, deriv

Possible outputs (i.e. "answers")

Regression: predicted average soil moisture level, Clas

Type of problem (e.g. classification, regression, recommendation...)

Regression and Classification

Baseline: simple, alternative way of making predictions (e.g. manual rules)

Regression: district-wise historical average.
Classification: majority class assignment

Performance evaluation

Domain-specific / bottom-line metrics for monitoring performance in production

Inform irrigation schedules, avoid drought damage.

Prediction accuracy metrics (e.g. MSE if regression; % accuracy, #FP for classification)

Regression: MAE, RMSE, R^2.

Classification: Accuracy, F1-score, Confusion Matrix.

Offline performance evaluation method (e.g. cross-validation or simple training/test split)

Temporal split (train on 2018, test on 2020)
Cross-validation for hyperparameter tuning.

Data preparation

How do we get training data (inputs, and outputs if supervised learning)? How many data points?

Inputs: Date, State, District, derived features from all over the India
Outputs: Avg Soil Moisture (Regression), Category (Classification).
~2.5 lakh rows.

Input features (extracted from data sources). If too many, list types of features and mention key ones.

Original: Date, State, District, Avg Soil Moisture Level, Avg Soil Moisture Volume, Aggregate %, Volume %.

Derived: Month, DayOfYear, lag features, rolling averages, moisture category

Using predictions

When do we make predictions and how many?

Predictions for upcoming months or specific planning dates.

Can be batch predictions (daily/weekly) or on-demand.

What is the time constraint for making those predictions?

No strict real-time constraint; dashboards updated daily/weekly/monthly.

How do we use predictions and confidence values?

Display on interactive dashboard (Plotly Dash / Streamlit).
Highlight risk zones, recommend irrigation schedules, policy alerts.
Confidence values guide caution thresholds for decision-making.

Learning models

When do we create/update models? With which data / how much?

Initially trained on 2018 data; retrained with new yearly data (e.g., 2020).

What is the time constraint for creating a model?

Model updates can happen offline (hours), not real-time.

Criteria for deploying model (e.g. minimum performance value — absolute, relative to baseline or to previous model)

Regression: RMSE / MAE lower than baseline.

Classification: Accuracy / F1-score higher than baseline or previous model.

Data Science Canvas

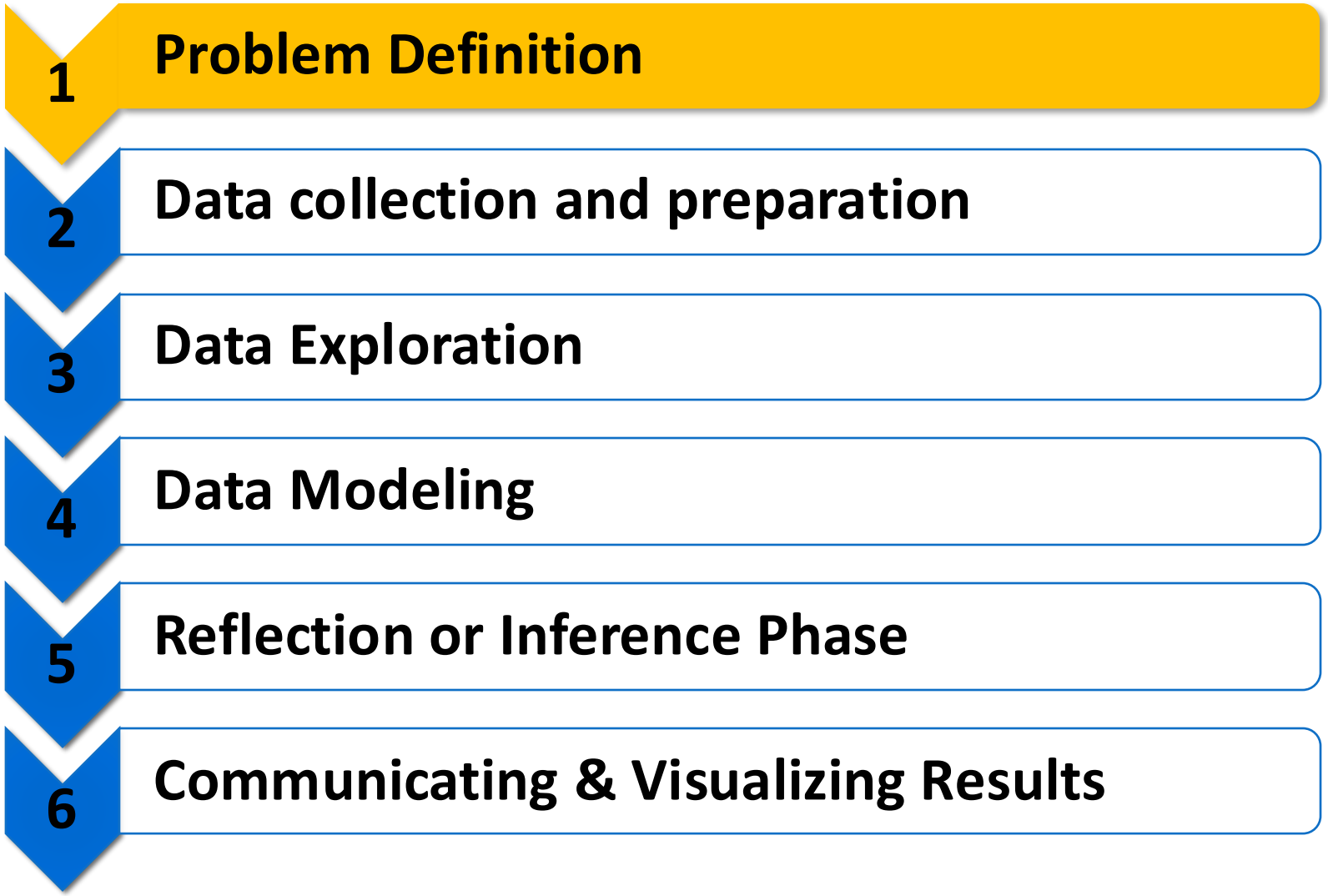
What is a Data Science Use Case Canvas?

A template that helps define the project's context, objectives, data, constraints, and deliverables. It's best to fill out the canvas during a brainstorming session with the project's stakeholders.

References:

- DS Canvas: https://github.com/tomalytics/datasciencecanvas
- ML Canvas: https://github.com/louisdorard/machine-learning-canvas
- ML Canvas – Churn prediction: https://github.com/louisdorard/machine-learning-canvas/blob/master/churn.pdf

Data Science Workflow



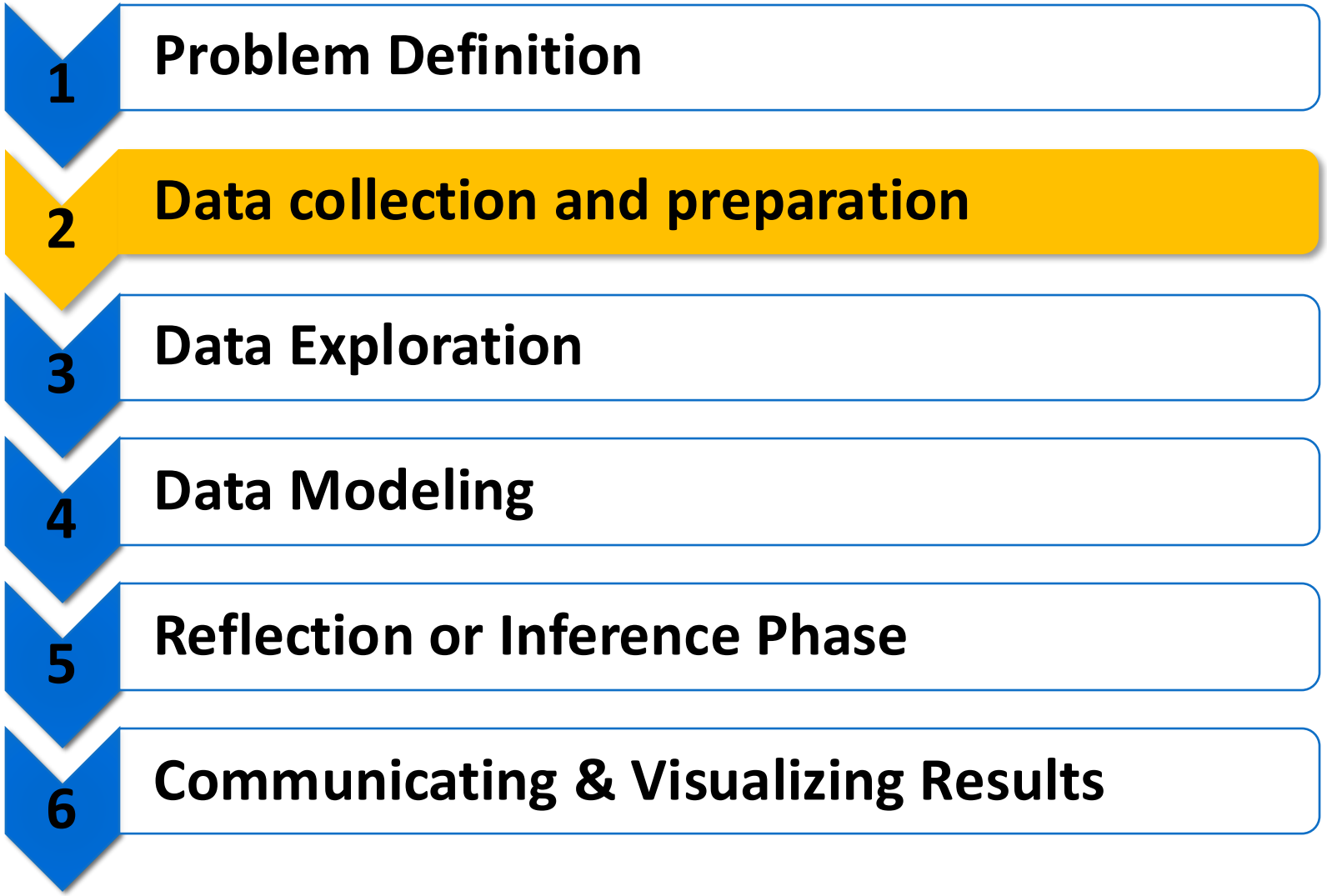
Step 1: Problem Definition

- Define the objective in business terms.
- How will your solution be used?
- What are the current solutions/workarounds (if any)?
- How should you frame this problem. (supervised/unsupervised, online/offline, etc.)?
- How should performance be measured?
- Is the performance measure aligned with the business objective?
- What would be the minimum performance needed to reach the business objective?
- What are comparable problems? Can you reuse experience or tools?
- Is human expertise available?
- How would you solve the problem manually?
- List the assumptions you (or others) have made so far
- Verify assumptions if possible.

Reducing Customer Churn for an Online Retailer

- **Objective in Business Terms:** Reduce customer churn by 10% within six months.
- **How Will Your Solution Be Used?** The marketing and customer success teams will use the solution to target at-risk customers with retention strategies.
- **Current Solutions/Workarounds:** The company uses basic heuristics to identify potential churners, but this approach has limited accuracy.
- **How Should You Frame This Problem?** It's a **supervised learning** problem, predicting churn based on historical data, and is handled **offline**.
- **How Should Performance Be Measured?** Use **precision, recall, F1-score**, and **AUC-ROC** to balance accuracy in identifying churners.
- **Is the Performance Measure Aligned with the Business Objective?** Yes, as it ensures resources are focused on true at-risk customers while maximizing retention.
- **What Would Be the Minimum Performance Needed?** Aim for at least 80% precision and recall to meet the churn reduction goal.
- **What Are Comparable Problems?** Churn prediction models from telecommunications and subscription services can be adapted.
- **Is Human Expertise Available?** Yes, the customer success team can provide insights for feature selection and model validation.
- **How Would You Solve the Problem Manually?** Identify patterns in past churners' behavior and target similar customers with retention offers.
- **List the Assumptions:**
  - Churn can be predicted from historical data.
  - Targeted interventions can effectively retain customers.
  - Available data is sufficient and reliable.
  - The model will generalize to future behavior.
- **Verify Assumptions:** Run preliminary analyses and small-scale tests to confirm predictions and intervention effectiveness.

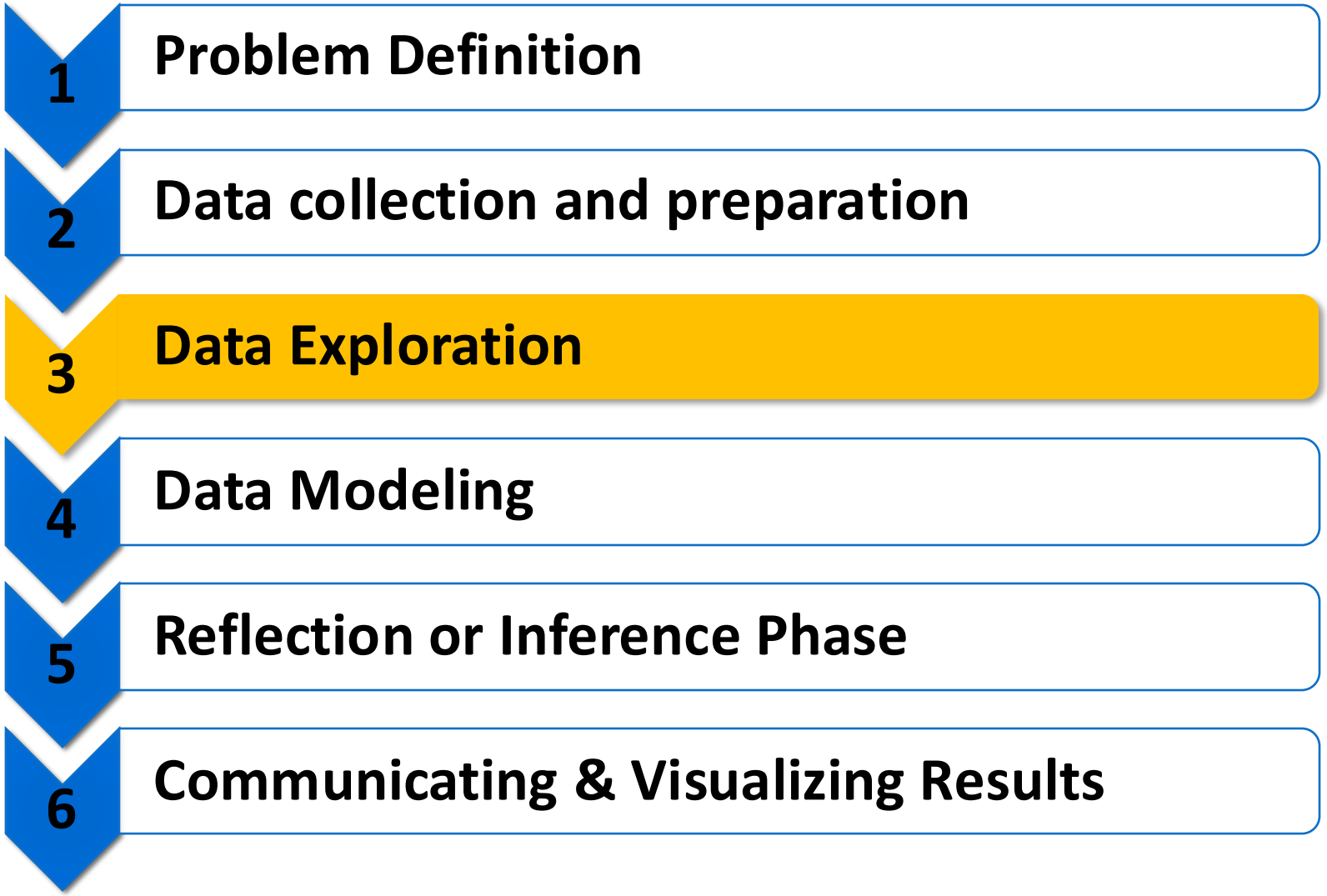
Data Science Workflow



Step 2: Data Collection and Preparation

- List the data you need and how much you need.
- Find and document where you can get that data.
- Check how much space it will take.
- Check legal obligations and get authorization if necessary.
- Get access authorization.
- Create a workspace (with enough storage space).
- Get the data.
- Convert the data to a format you can easily manipulate (without changing the data itself).
- Ensure sensitive information is deleted or protected (e.g., anonymized).
- Check the size and type of data (time series, sample, geographical, etc.).
- Sample a test set, put it aside, and never look at it (no data snooping).

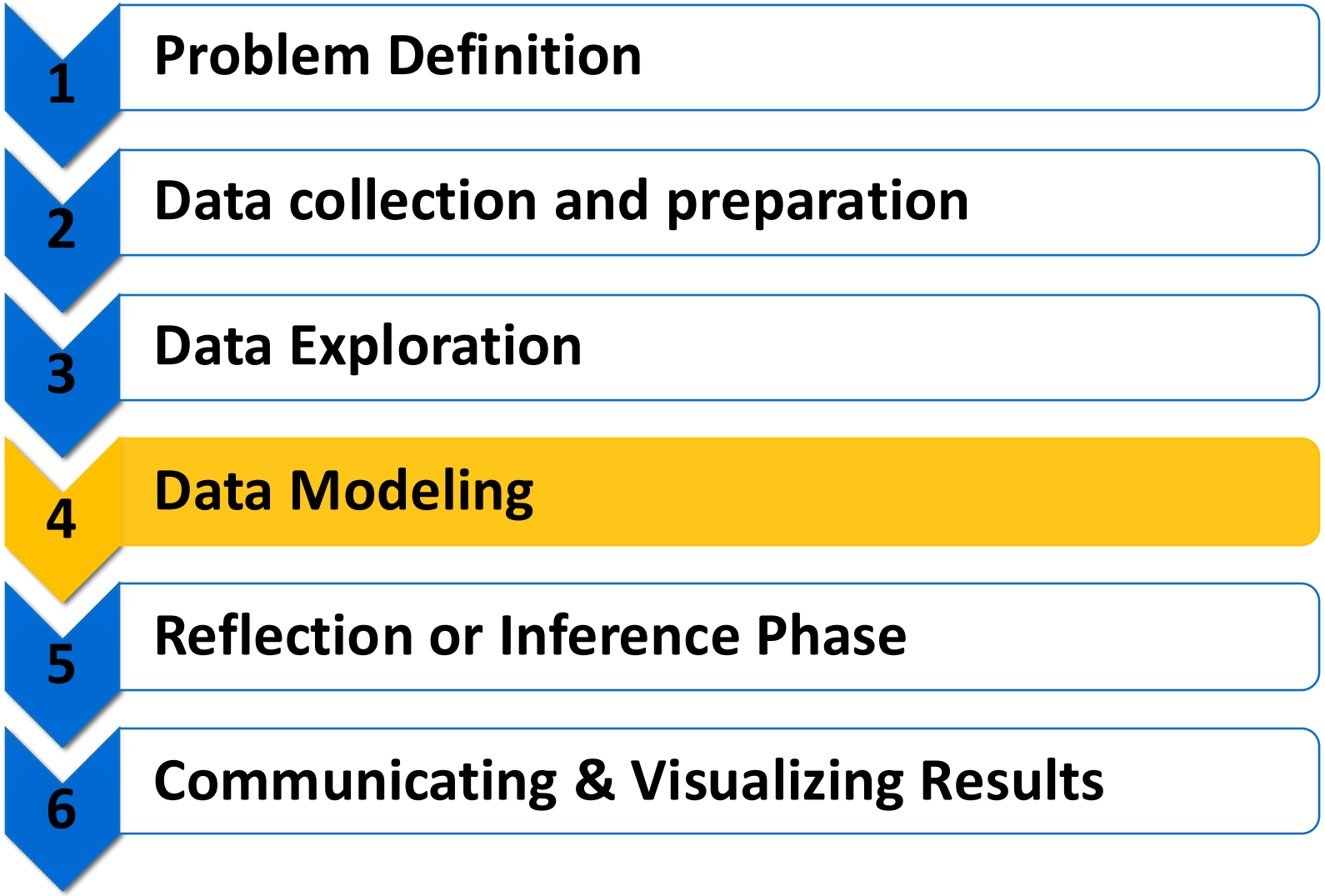
Data Science Workflow



Step 3: Data Exploration

- Create a copy of the data for exploration (sampling it down to a manageable size if necessary).
- Create a Jupyter notebook to keep a record of your data exploration
- Study each attribute and its characteristics:
  - Name and Type (categorical, int/float, bounded/unbounded, text, structured, etc.)
  - % of missing values
  - Noisiness and type of noise (stochastic, outliers, rounding errors, etc.)
  - Usefulness for the task
  - Type of distribution (Gaussian, uniform, logarithmic, etc.)
- For supervised learning tasks, identify the target attribute(s).
- Visualize the data.
- Study the correlations between attributes.
- Study how you would solve the problem manually.
- Identify the promising transformations you may want to apply.
- Identify extra data that would be useful.
- Document what you have learned.

Data Science Workflow



Model Development

- Train many quick and dirty models from different categories (e.g., linear, naive, Bayes, SVM, Random Forests, neural net, etc.) using standard parameters.
- Measure and compare their performance.
- For each model, use N-fold cross-validation and compute the mean and standard deviation of their performance.
- Analyze the most significant variables for each algorithm.
- Analyze the types of errors the models make.
- What data would a human have used to avoid these errors?
- Have a quick round of feature selection and engineering.
- Have one or two more quick iterations of the five previous steps.
- Short-list the top three to five most promising models, preferring models that make different types of errors.