Best Practices for Competition Success

STEP 1: UNDERSTAND THE PROBLEM STATEMENT

- **Clarify Objectives & Business Impact**: Determine whether it's a classification or regression task, the domain context, and what "success" looks like. ✓
- **Target Variable**: Understand how the target is defined and distributed. ✓
- **Constraints & Permissions**: External data limits, privacy, or regulatory issues. ✓
- **Action Points**: ✓
    - Annotate key requirements and assumptions. ✓
    - Research the domain for feature ideas. ✓
    - Restate the problem in your own words to ensure clarity. ✓

STEP 2: STUDY THE EVALUATION METRIC

- **Metric Matters**: Each competition or project has a specific metric (e.g., F1-score, AUC-ROC, RMSE), which guides how you tune and compare models. ✓
- **Classification vs. Regression Metrics**:
  - Accuracy, F1-score, AUC-ROC, and Log Loss for classification.
  - RMSE, MAE, R^2 for regression.
- **Optimization Direction**: Decide if you must maximize or minimize the metric; some metrics (e.g., AUC-ROC) may need a proxy loss for training.
- **Action Points**:
  - Simulate prediction changes to see impact on the metric.
  - If allowed, implement custom losses aligned to the competition metric.

STEP 3: PERFORM AN INITIAL DATA ANALYSIS (EDA)

- **Data Profiling**: Summarize shape, basic statistics, feature and target distribution.
- **Quality Checks**: Identify missing values, outliers, or suspicious patterns; confirm no hidden data leakage.
- **Train-Test Shift**: Check if train and test set differ significantly in their feature distributions.
- **Action Points**:
  - Use tools like Pandas Profiling or Sweetviz for automated EDA.
  - Document anomalies or domain-specific oddities.
  - Examine correlations, scatter plots, and histograms.

STEP 4: DEVELOP A BASELINE MODEL

- **Why a Baseline?** A simple model (e.g., Logistic Regression, Decision Tree) gives you a reference performance. ✓
- **Pipeline Check**: Make sure that data loading, preprocessing, and validation splits are correct. ✓
- **Diagnostics** : Analyze misclassifications/ residuals to spot improvement areas.
- **Action Points**:
  - Start with a naive or default-parameter model. ✓
  - Use cross-validation to assess reliability. ✓
  - Record baseline metrics to quantify future gains. ✓

STEP 5: SET UP A ROBUST VALIDATION STRATEGY

- **Importance of Validation**: Avoid overfitting or a single random split.
- **Common Methods**:
  - K-Fold (balanced data).
  - Stratified K-Fold (imbalanced classes).
  - Time-Based or Group Splits (time series, repeated measures).
- **Action Points**:
  - Match validation approach to the final test scenario. ✓
  - Keep a holdout set for final checks. ✓
  - Use consistent folds across experiments to compare fairly. ✓

STEP 6: FEATURE ENGINEERING AND DATA PREPROCESSING

- **Missing Data**: Mean/median imputation, advanced methods, or flags. ✓
- **Encoding Categorical Variables**: One-hot/ target/ frequency enc. (leakage?!). ✓
- **Scaling & Transformations**: Normalize or log-transform skewed features, especially for sensitive models (SVMs, neural nets). ✓
- **New Features**: Time-based, domain knowledge, interactions (e.g. ratios).
- **Action Points**:
  - Add features in small batches and track validation changes.
  - Use scikit-learn Pipelines to avoid leakage.
  - Consider dimensionality reduction if feature space is large.

STEP 7: SELECT AND TUNE MODELS STRATEGICALLY

- **Model Choices**: ✓
  - Tree-based ensembles (XGBoost, LightGBM, CatBoost) for tabular data.
  - Neural networks for large-scale or unstructured data.
  - Simple linear/logistic models can still compete with good feature engineering.
- **Hyperparameter Tuning**:
  - Grid Search, Random Search, Bayesian Optimization. ✓
  - Early stopping and regularization (L1, L2) to avoid overfitting.
- **Action Points**:
  - Track experiments (MLflow, Weights & Biases, or spreadsheets).
  - Start with defaults and refine gradually.
  - Use IML tools (SHAP, permutation importance) for feature insights.

STEP 8: TRACK EXPERIMENTS AND ITERATE

- **Experiment Logging**: Keep detailed records of data versions, hyperparams, code revisions. ✓
- **Error Analysis**: Study misclassifications and residuals to guide feature tweaks.
- **Iterative Improvements**:
  - Refine features, adjust validation, revisit transformations.
  - Maintain a “changelog” for each iteration.
- **Action Points**:
  - Automate repeated tasks (submissions, data prep). ✓
  - Prune ineffective ideas swiftly. ✓
  - Ensure reproducibility (version control, environment snapshots). ✓

STEP 9: UNDERSTAND THE LEADERBOARD AND AVOID OVERFITTING

- **Public vs. Private Leaderboard**: Public boards use only part of the test set—overfitting leads to dramatic rank changes later. ✓
- **Submission Strategy**:
  - Rely mainly on internal validation. ✓
  - Keep a private holdout to confirm final performance. ✓
- **Leaderboard Shakeups**: Watch for big changes when private scores are revealed; it often indicates overfitting to the public set. ✓
- **Action Points**:
  - Limit minor tweaks solely to improve a public rank.
  - If a jump seems suspicious, re-check data handling for leakage.

STEP 10: COLLABORATE AND LEARN FROM OTHERS

- **Study Top Solutions**: Many winning approaches are documented in blog posts or Kaggle discussions.
- **Teamwork**:
  - Collaboration brings diverse perspectives and model ensembling.
  - Share code and features for synergy.
- **Community & Networking**:
  - Discussion forums reveal dataset quirks, best practices.
  - Local meetups or online communities keep you updated.
- **Action Points**:
  - Document and share your findings.
  - Adapt others' ideas carefully—validate them on your own splits.
  - Engage with peers for new tools and techniques.

FINAL ADVICE

- **Start Simple, Then Iterate**: Don't jump into complex models, get a solid baseline.
- **Prioritize Validation**: A robust validation strategy prevents nasty surprises.
- **Log Everything**: Document transformations, parameters, and model versions.
- **Aim for Generalizable Solutions**: Don't just chase leaderboard scores—focus on reproducibility and reliability.