1. **Frame Problem – Look at Big Picture**
   1. Define objective in business terms:
   2. How will the solution be used?
   3. What are the current solutions/workarounds (if any)?
   4. How will this problem be framed (supervised/unsupervised, online/offline, etc.)?
   5. How should performance be measured?
   6. Is the performance measure aligned with business objective?
   7. What is the minimum performance needed to reach the business objective?
   8. Are there comparable problems and can we reuse experience or tools?
   9. Is human expertise available?
   10. How is the problem solved manually?
   11. List the assumptions we’ve made so far:
   12. Verify assumptions if possible.
2. **Obtain Data**
   1. List the data you need and how much you need.
   2. Find and document where you can get that data.
   3. How much space it will take?
   4. Check legal obligations and get authorization if necessary.
   5. Get access authorizations.
   6. Create a workspace (with enough storage space).
   7. Get the data.
   8. Convert the data to a format that is easily manipulate (without changing the data itself).
   9. Ensure sensitive information is deleted or protected (e.g., anonymized).
   10. Check the size and type of data (time series, sample, geographical, etc.).
   11. Sample a test set, put it aside, and never look at it.
3. **Explore Data (Use Field Experts)**
   1. Create a copy of the data for exploration (sampling it down to a manageable size if necessary).
   2. Create a Jupyter notebook to keep a record of your data exploration.
   3. Study each attribute and its characteristics:
      1. Name
      2. Type (categorical, int/float, bounded/unbounded, text, structured, etc.)
      3. % of missing values
      4. Noisiness and type of noise (stochastic, outliers, rounding errors, etc.)
      5. Possibly useful for the task?
      6. Type of distribution (Gaussian, uniform, logarithmic, etc.)
   4. For supervised learning tasks, identify the target attribute(s).
   5. Visualize the data.
   6. Study the correlations between attributes.
   7. Study how you would solve the problem manually.
   8. Identify the promising transformations you may want to apply.
   9. Identify extra data that would be useful.
   10. What have we learned so far:
4. **Prepare Data**
   1. Work on copies of the data (keep the original dataset intact).
   2. Write functions for all data transformations.
   3. Data cleaning:
      1. Fix or remove outliers (optional).
      2. Fill in missing values (e.g., with zero, mean, median…) or drop their rows (or columns).
      3. Feature selection (optional):
         1. Drop the attributes that provide no useful information for the task.
      4. Feature engineering, where appropriate:
         1. Discretize continuous features.
         2. Decompose features (e.g., categorical, date/time, etc.).
         3. Add promising transformations of features (e.g., log(x), sqrt(x), x^2, etc.).
         4. Aggregate features into promising new features.
      5. Feature scaling: standardize or normalize features.
5. **Short-List Promising Models**
   1. Train quick and dirty models from different categories (linear, naïve Bayes, SVM, Random Forests, neural net, etc.) using standard parameters.
   2. Measure and compare their performance.
      1. For each model, use N-fold cross-validation and compute the mean and standard deviation of the performance measure on the N folds.
   3. Analyze the most significant variables for each algorithm.
   4. Analyze the types of errors the models make.
      1. What data would a human have used to avoid these errors?
   5. Perform a quick round of feature selection and engineering.
   6. Have one or two more quick iterations of the five previous steps.
   7. Short-list the top three to five most promising models, preferring models that make different types of errors.
6. **Fine-Tune the System**
   1. Fine-tune the hyperparameters using cross-validation.
      1. Treat data transformation choices as hyperparameters, especially when unsure about them (e.g., impute missing values with zero or with the median value? Or drop rows?)
      2. Unless there are very few hyperparameter values to explore, prefer random search over grid search. If training is very long, use Bayesian optimization approach (e.g. Gaussian process priors)
   2. Try Ensemble methods.
      1. Combining your best models will often perform better than running them individually.
   3. If confident about final model, measure performance on the test set to estimate the generalization error.
      1. **DO NOT** tweak model after measuring the generalization error (yields to overfitting test set)
7. **Present Solution**
   1. Document what has been done.
   2. Create awesome presentation.
      1. Make sure you highlight the big picture first.
   3. Explain why your solution achieves the business objective.
   4. Don’t forget to present interesting points you noticed along the way.
      1. Describe what worked and what did not.
      2. List your assumptions and your system’s limitations.
   5. Ensure your key findings are communicated through beautiful visualizations or easy-to-remember statements
8. **Launch** 
   1. Get solution ready for production (plug into production data inputs, write unit tests, etc.).
   2. Write monitoring code to check your system’s live performance at regular intervals and trigger alerts when it drops.
   3. Beware of slow degradation: models tend to “rot” as data evolves
      1. Measuring performance may require a human pipeline (e.g., via a crowdsourcing or manual human validation).
      2. Monitor inputs quality (e.g., a malfunctioning equipment causing random values, or another team’s output becoming stale).
         1. This is very important for online learning systems.
   4. Retrain your models on a regular basis on fresh data (automate as much as possible).