Machine Learning Project Checklist

Frame the problem and look at the big picture

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

Get the Data

Note: Automate as much as possible so you can easily get fresh data

(Only automate if the project will go into production or there is a need for repetition.)

1. List the data you need and how much you need.
2. Find and document where you can get that data
3. Check 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 you can 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 (no data snooping!).

Explore the Data

Note: Try to get insights from a field expert for these steps

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:
    - Name
    - Type (categorical, int/float, bounded/unbounded, text, structured, etc.)
    - % of missing values
    - Noisiness and type of noise (stochastic, outliers, rounding errors, etc.)
    - Possibly useful for the task?
    - 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 (go back to “Get the Data”).
10. Document what you have learned.

Prepare the Data

Notes:

- Work on copies of the data (keep the original dataset intact).
- Write function for all data transformations you apply, for five reasons:
    - So you can easily prepare the data the next time you get a fresh dataset
    - So you can apply these transformations in future projects
    - To clean and prepare the test set
    - To clean and prepare new data instances once your solution is live
    - To make it easy to treat your preparation choices as hyperparameters

1. Data cleaning
    - Fix or remove outliers (optional)
    - Fill in missing values (e.g., with zero, mean, median…) or drop their rows (or columns)
2. Feature selection (optional)
    - Drop the attributes that provide no useful information for the task
3. Feature engineering, where appropriate
    - Discretize continuous features
    - Decompose features (e.g., categorical, date/time, etc.)
    - Add promising transformations of features (e.g., log(x), sqrt(x), x^2, etc.)
    - Aggregate features into promising new features
4. Feature scaling: standardize or normalize features

Short-list promising models

Notes:

- If the data is huge, you may want to sample smaller training sets so you can train many different models in a reasonable time (be aware that this penalizes complex models such as large neural nets or Random Forests).
- Once again, try to automate these steps as much as possible.

1. Train many quick and dirty models from different categories (e.g., linear, naïve Bayes, SVM, Random Forests, neural net, etc.) using standard parameters
2. Measure and compare their performance
    - 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
    - What data would a human have used to avoid these errors?
5. Have 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.

Fine-tune the system

Notes:

- You will want to use as much data as possible for this step, especially as you move toward the end of fine-tuning.
- As always automate what you can.

1. Fine-tune the hyperparameters using cross-validation
    - Treat your data transformation choices as hyperparameters, especially when you are not sure about them (e.g., should I replace missing values with zero or with the median value? Or just drop the rows?).
    - Unless there are very few hyperparameter values to explore, prefer random search over grid search. If training is very long, you may prefer a Bayesian optimization approach (e.g., using Gaussian process priors, as described by Jasper Snoek, Hugo Larochelle, and Ryan Adams (http://homl.info/134)).
2. Try Ensemble methods. Combining your best models will often perform better than running them individually.
3. Once you are confident about your final model, measure its performance on the test set to estimate the generalization error.

Important: Don’t tweak your model after measuring the generalization error: you would just start overfitting the test set.

Present your solution

1. Document what you have done
2. Create nice presentation
    - 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
    - Describe what worked and what didn’t.
    - List your assumptions and your system’s limitations.
5. Ensure your key findings are communicated through beautiful visualizations or easy-to-remember statements (e.g., “the median income is the number-one predictor of housing prices”).

Launch!

1. Get your 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.
    - Beware of slow degradation too: models tend to "rot" as data evolves.
    - Measuring performance may require a human pipeline (e.g., via a crowdsourcing device).
    - Also monitor your inputs' quality (e.g., a malfunctioning sensor sending random values, or another team's output becoming stale). This is particularly important for online learning systems.
3. Retrain your models on a regular basis on fresh data (automate as much as possible).
4. If in production, automate as much as possible.

➔ For each step, break "features" (functions) into modules to allow easier automation.
➔ Consider writing custom Python libraries
➔ Build a project environment with a dependencies file