Data Science Project Checklist (adapted from Aurélien Géron's book *Hands-On Machine Learning with Scikit-Learn TensorFlow*)

Andreas W. Kempa-Liehr

May 30, 2017

Contents

1 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.

2 Get the data

Automate as much as possible so you can easily get fresh data.

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. Check the size and type of data (time series, sample, geographical, etc.).
10. Ensure sensitive information is deleted or protected.
11. Sample a test set, put it aside, and never look at it.
12. Check your workspace setup.

3 Explore the data

Try to get insights from a domain expert for these steps.

1. Create a Jupyter notebook to keep a record of your data exploration.
2. Study each attribute and its characteristics.
3. For supervised learning tasks, identify the target attribute(s).
4. Visualize the data.
5. Study the correlations between attributes.
6. Engineer features and identify useful transformations.
7. Study how you would solve the problem manually.
8. Identify extra data that would be useful.
9. Document what you have learned.

4 Prepare the data

Work on copies of the data (keep the original dataset intact).
Write functions for all data transformations you apply

1. Data Cleaning
   - Fix or remove outliers (optional).
   - Fill in missing values
2. Decompose features
   - categorical
   - date/time
3. Feature selection (optional)
   - Drop the attributes that provide no useful information for the task
4. Feature engineering, where appropriate
   - Discretize continuous features.

- Add promising transformations of features
- Aggregate features into promising new features

5. Feature scaling: standardize or normalize features.

5 Short-List Promising Models

If the data is huge, you may want to sample smaller training sets so you can train many different models in a reasonable time

1. Train many quick and dirty models from different categories 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.
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.

6 Present Your Solution

1. Document what you have done.
2. Create a 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 did not.

- 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").

7 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 service).
   - 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).