Tackling a Machine Learning Project: 3 Main Steps

1. Identify a suitable problem

This is probably the hardest part. Get creative! (Usually, you would brainstorm together with ML and domain experts.)

Motivation

Situation / Problem / Goal
Which process or task could be automated? What situation could be improved with more insights / by better planning? I.e., where do you see a lot of inefficiencies around you that could be mitigated by a better use of data? For example, you could look for opportunities to decrease wasted resources / time / costs or increase revenue / customer satisfaction / etc.

Business Key Performance Indicator (KPI, i.e., what to optimize)
With what metric can you measure/quantify the progress towards your goal?

Status Quo (& Target)
What is the value of this KPI right now? And what could be considered an ambitious yet realistic target in terms of improving the situation, i.e., when are you 'done'?

Value Generation
How could the company make money on this or reduce costs? Could it improve an internal process (e.g., maybe a process can be run more efficiently with the insights from an analysis or a tedious task that would otherwise require a human worker can be automated using an ML model)? Could the ML model be integrated as a new feature within an existing product and thereby, e.g., make this product more appealing to customers? Could the ML solution be sold as an entirely new product, e.g., offered as a Software-as-a-Service (SaaS) solution?
Please note that how the ML solution will be used in the end might also be a strategic decision that can be different for every company. For example, an ML solution that recognizes scratches in produced products might be used by one company to improve their internal production process, while another company that produces the machines that make the products could integrate this as a new feature in their machines, and a third company might offer this as a SaaS solution compatible with different production lines.

Solution Outline

Deliverables
Does the solution consist of a piece of software that is to be deployed somewhere to continuously make predictions for new data points, or are you more interested in the insights gained from an one-off analysis?

Workflow Integration
Especially important for a software project: Where does the ML solution fit within the rest of the infrastructure, i.e., where do you get the input data from and where do you need to send the results to?

1 Data Point & Inputs
What is one sample / observation and what kind of input features does the ML model receive for it?

Type of ML Solution & Output
Which category of ML algorithms produces the desired output for each data point? (See chapter "ML Solutions: Overview".)

Build or Buy?
Does the solution require unique domain knowledge only available at your company, e.g., because you're analyzing data generated by your own specific processes/machines and/or will the solution be a key part of your business, e.g., a new feature that makes your products more attractive? Or is this a common (but complex) problem, for which a solution already exists (e.g., offered as a Software-as-a-Service (SaaS) product), that you could buy off the shelf? For example, extracting the relevant information from scanned invoices to automate bookkeeping processes is a relatively complex task for which many good solutions already exist, so unless you are working in a company building bookkeeping software and plan to sell a better alternative to these existing solutions, it probably doesn't make sense to implement this yourself.
Here are some general points you might want to consider when deciding whether to buy an ML solution or build it yourself:

- How much effort would be required in terms of preprocessing your data before you could use the off-the-shelf ML solution?
- How difficult would it be to integrate the output from the off-the-shelf ML solution into your general workflow? Does it do exactly what you need or would additional post-processing steps be required?
- How reliable is the off-the-shelf ML solution? Are there any benchmarks available and/or can you test it with some common examples and edge cases yourself?
- How difficult would it be to implement the ML solution yourself? For example, what kind of open source libraries exist that solve such a task? Do you have the necessary ML talent or would you need to hire, e.g., freelancers?

- Can the off-the-shelf ML solution be deployed in-house or does it run on an external server and would this bring with it any data privacy issues?
- How high are the on-going licensing fees and what is included in terms of maintenance (e.g., how frequently are the models retrained)?

Unless the ML solution will be an integral part of your business, in the end it will probably come down to comparing costs for building, running, and maintaining the system yourself vs. costs for integrating the off-the-shelf solution into your existing workflow (incl. necessary data preprocessing) and on-going licensing fees.

Challenges & Risks

Data Availability (Quality & Quantity)
How good do you think your data is? (You might want to reread the section on Data: 'Garbage in, garbage out')
How much data do you have (including rare events)? If the answer is "None", start collecting *right now!*
Do you already see some problems with the data that you should take into account later, e.g., systematic biases?

⇒ *Next steps to get / improve data?*
Getting the necessary data is always the first step in creating an ML solution. Who would you talk to, to get this data? Who would you talk to, to set up / improve the data infrastructure and what should be the next steps to improve data quality and quantity?

Is an ML solution feasible?
What does a human expert think about this problem, i.e., if she was presented with the input, could she identify the correct output? This does not necessarily apply to unsupervised learning problems, but even then the intuition of an expert can give valuable hints!
Depending on the expert's answer, proceed as follows:

- the expert tells you the problem is trivial to solve
  → ask the expert to come up with some rules / equations and automate this the traditional way (i.e., without ML)
- the expert can easily solve the problem, but can't explain how she did it
  → this sounds like a perfect use case for ML!
- the expert can in principle solve the problem, but is sometimes unsure which output is the correct one
  → try to define your problem more clearly and come up with less ambiguous output options (i.e., labels)
- the expert can't solve the problem with the given inputs, but thinks that it should in principle be possible
  → can you get additional or less noisy inputs?
- the expert thinks your idea is crazy
  → you can still try ML (the results might surprise you!), but your time might be better spent on a different problem

Bonus: Has a similar problem been solved with ML before?
If you need to devise a novel neural network architecture to solve the problem, for example, because you are dealing with super fancy input and/or output data types (think AlphaFold), this will probably require years of research. Your chances of success are much higher, if you can use some existing algorithm that has already been used to solve a similar problem.

What else could go wrong (e.g., legal issues / ethical concerns)?
Are there any concerns w.r.t. data privacy? What about accountability, i.e., do the decisions of the machine learning model need to be transparent, for example, if someone is denied credit because of an algorithmically generated credit score? Why might users get frustrated with the solution?

⇒ *Reduced Risk Rollout: How could the risks be mitigated?*
Your ML system (like humans) will make mistakes. This is especially true since your input data will probably change over time and users might even try to intentionally deceive the system (e.g., spammers come up with more sophisticated messages if their original ones are caught by the spam filter). What would be the worst case scenario and how much risk are you willing to take? Instead of going all in with ML from day 1, is there a way your system can be monitored in the beginning while still providing added value?

2. Devise a working solution

You came up with a well defined input-output problem, you have collected a lot of high-quality data, a domain expert gave you the green light, and management is impressed by the low risk and immediate value your solution would provide. Now it's time to get your hands dirty!

Of course, in reality, arriving at a working solution is an iterative process where you need to try many different models, tune hyperparameters, etc., so without programming we don't get very far here (this is the topic of the case study and cheat sheet). But you can still think about the first steps that you would take to solve your problem.

From Raw Data to Good Data
What does your raw data look like? Do you need to perform any extra steps to arrive at meaningful numeric features?
What would be the entries of the d-dimensional input feature vector representing one data point?
(Of course, this might change later on, for example, if you discover that you need to do more feature engineering, i.e., construct some additional, more complex features to improve the performance.)

ML Solution
What specific ML models could be used to solve your problem? What would be a simple baseline and which more complex models could you try next?

ML KPI, i.e., evaluation metric
What would be a suitable metric to evaluate the ML solution? Are some errors more critical than others?
What level of performance would be reasonable to expect (e.g., what is the human level performance on this task)?

3. Get it ready for production

Let's assume the model you've chosen has a decent performance on the historical data you've collected.
Now it's time to get your solution into production (which requires solving a set of additional challenges, for which you might want to consult with a data engineer and look into MLOps strategies), and/or to present the insights you've derived to the project's stakeholders:

ML Software Project:
Deployment: Cloud or Edge? Typically, a model is either run 'in the cloud' (i.e., on a central server that can handle more complex computations efficiently) or 'on the edge' (i.e., on each individual machine where the data is produced, which can be necessary if the data needs to be processed in real time and/or a continuous internet connection can not be guaranteed, e.g., in self-driving cars).
Continuous Monitoring: Don't forget that the model needs to be continuously monitored, incl. the collection of new data and frequent retraining of models to counteract data and concept drifts.

Data Science Insights Project:
Make sure to consult with a domain expert, especially when interpreting the model and explaining its predictions, to ensure it captures some true causal influences and does not rely on spurious correlations.