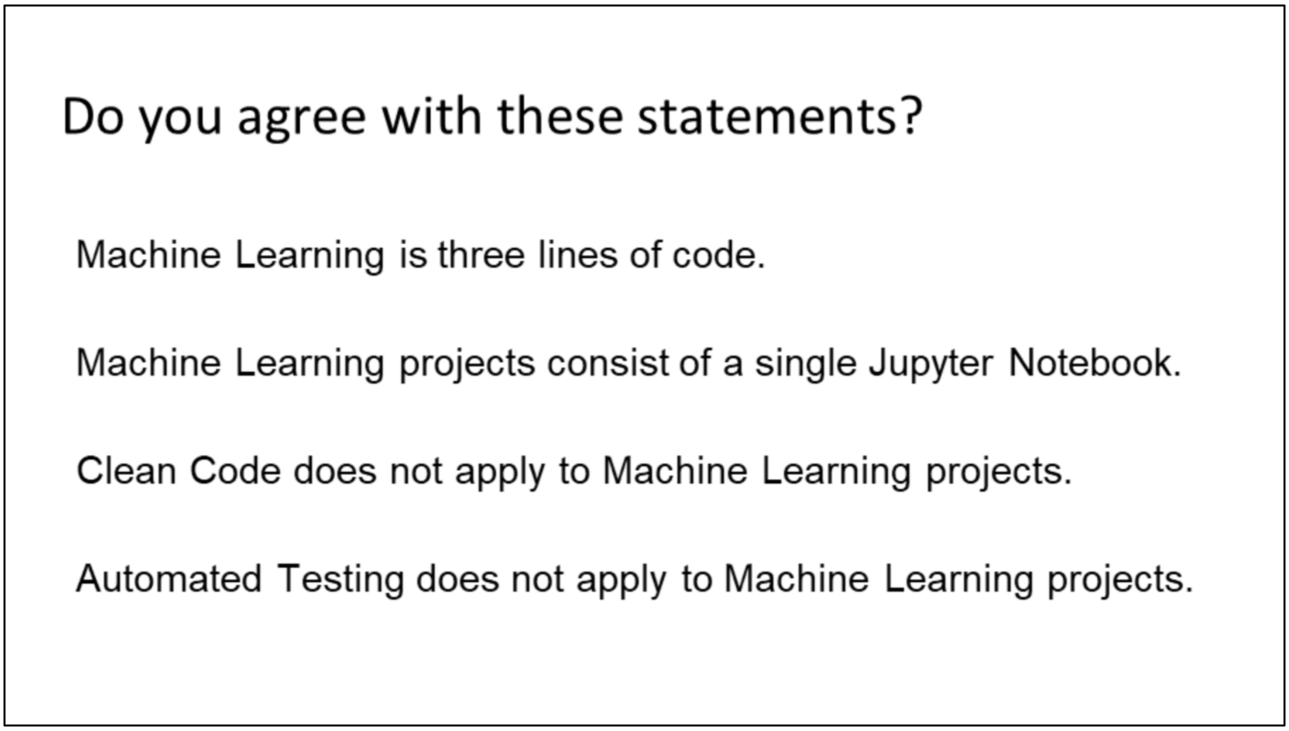
Complex Machine Learning Projects

Part 1: Machine Learning, I choose you!



Do you agree with these statements?

Machine Learning is three lines of code.

Machine Learning projects consist of a single Jupyter Notebook.

Clean Code does not apply to Machine Learning projects.

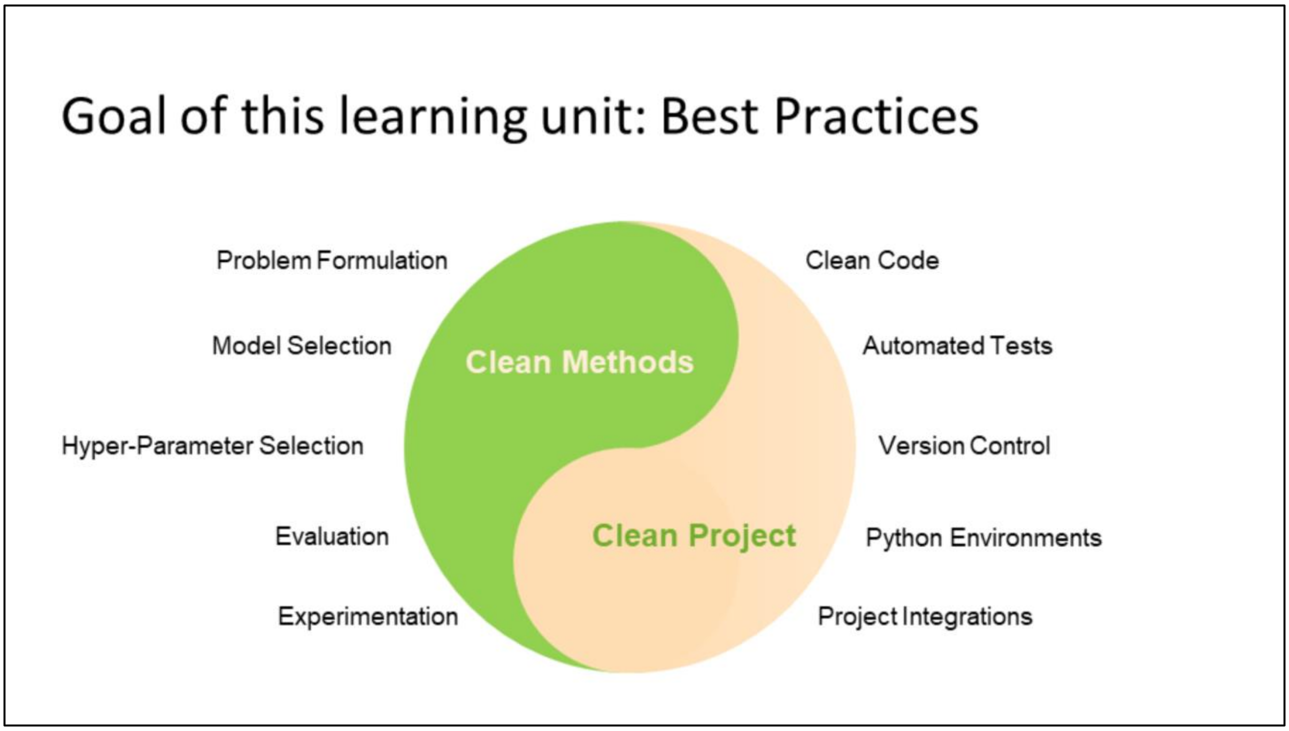
Automated Testing does not apply to Machine Learning projects.

Take a minute and reflect on these statements? Do you agree with them? Why could someone believe them? What issues could be caused by following them?

Of course, Machine Learning is not only three lines of code. It is true that machine learning models can be instantiated and learned in very few lines of code thanks to powerful libraries like Tensorflow and Keras. However, even with these libraries, the process of machine learning is often a lot more involved than just running a model. We need to explore and prepare data. Compare different models against each other. Define a good training and test setup. Find the right Hyper Parameters. And, of course, use the model after training it. While the code to instantiate and learn a model can be straightforward, the software system and processes that it is embedded in can by far from simple.

If you look up Machine Learning examples, you usually get a neat notebook containing all the steps of the example. That is one of the things notebooks are good for: creating a descriptive document that is integrated with code. However, that does not mean that all machine learning needs to be done in notebook environments or that notebooks can't be complemented with normal python modules. Machine learning works perfectly well outside of a notebook environment. Defining a notebook should be an active choice with a specific goal in mind and not a default because "this is what Machine Learners do".

Indeed, machine learning projects quickly exceed the scope that can comfortably be kept and presented in a single notebook. This is where you should naturally split up your code among different files, just the way you do with normal programming projects. And of course, once you have a project, you should aim to make it as readable and maintainable as possible. That’s where good software engineering practices, like clean code and automated testing come in.

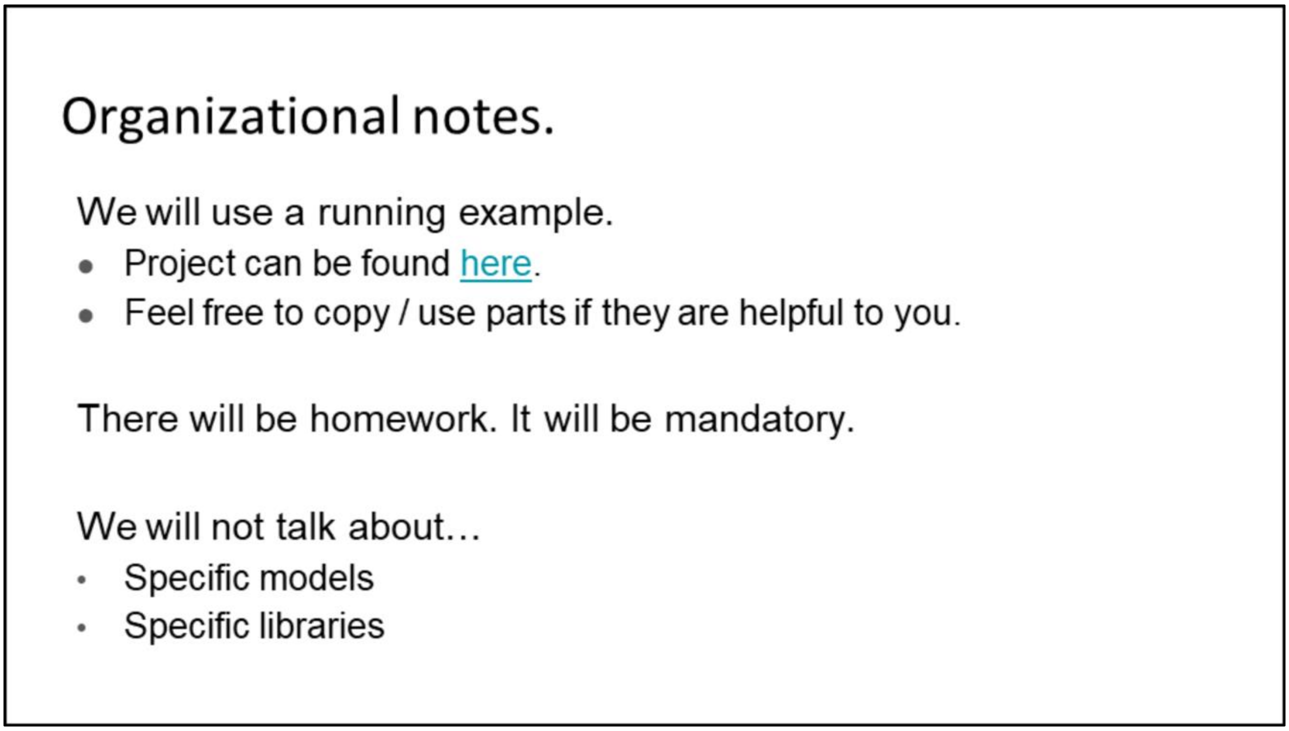


Goal of this learning unit: Best Practices

The goal of this learning unit is to help you make the transition from that introductory machine learning project in one single notebook to a more complex project.

We will look at this from two complementing sides:

- Clean Methods: this means applying clean methodology to your machine learning experiments. It involves all activities of the machine learning engineer, like problem formulation, model selection, hyper-parameter selection, and evaluation. A lot of this is going to involve defining executing and documenting experiments.
- Clean Projects: this means having a machine learning project that is easy to understand and maintain. It focuses on best practices in software engineering, such as clean code, automates testing, version control, etc. and how they apply to machine learning projects.



Organizational notes.

We will use a running example.

- Project can be found here.
- Feel free to copy / use parts if they are helpful to you.

There will be homework. It will be mandatory.

We will not talk about…

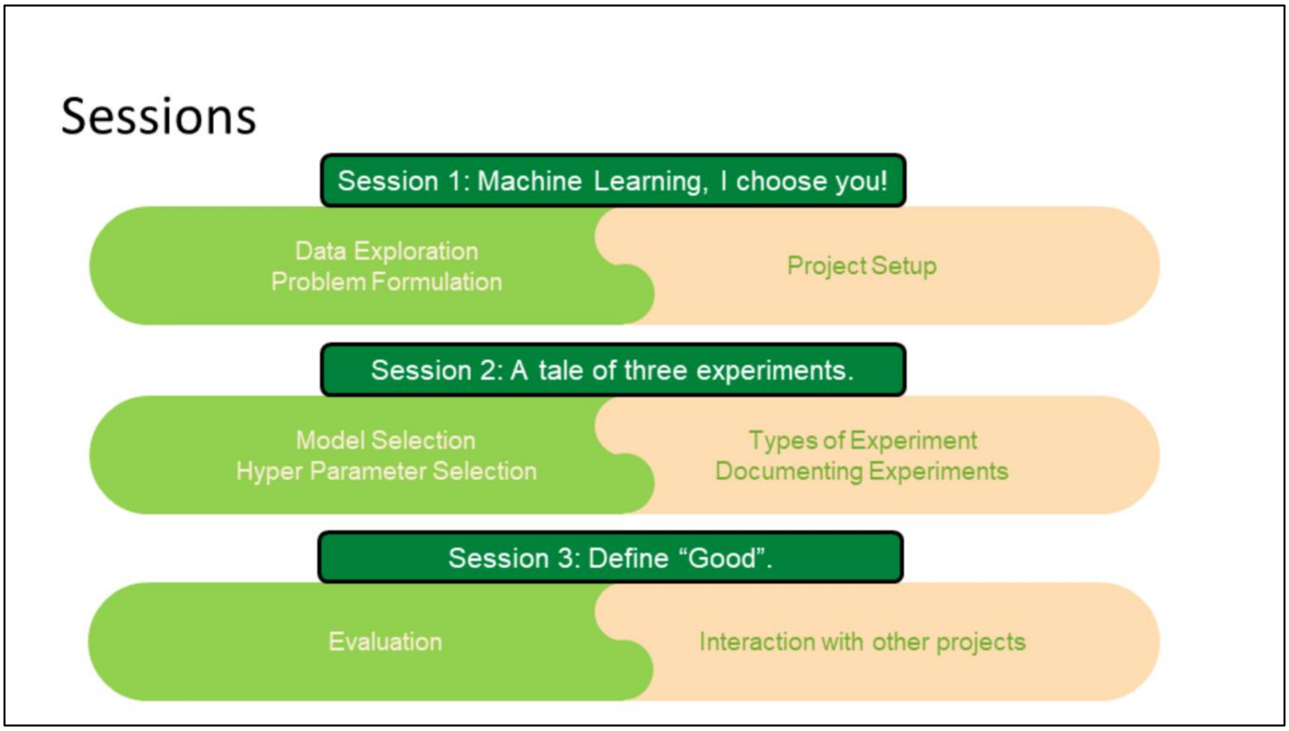
- Specific models
- Specific libraries

There are a few other organizational notes you should be aware of.

The project will be based on a running example. Based on a high-level goal, we will search for datasets, evaluate different models, select hyper parameters and evaluate our resulting model.
This is implemented in the linked git repository. Besides serving as a running example, this repository has been created as a resource for you to refer to. We will introduce the example later.
While looking through the project, you should be aware that we designed this project with the goal to be a good learning resource. This of course has influenced the design of the project. For example, some parts are a bit over-explained and contain more comments than strictly necessary.

The learning unit will contain some mandatory homework. The homework will be light touch (two to four hours maximum) and will be required so we all are on the same page in our discussions.

This learning unit has been designed to focus on best practices surrounding machine learning. We will not focus on specific models, the mathematics behind them or libraries you can use. There are good learning resources out there for these and you should pick whichever fit the problem you intend to solve.



Sessions

Session 1: Machine Learning, I choose you!

Data Exploration
Problem Formulation

Project Setup

Session 2: A tale of three experiments.

Model Selection
Hyper Parameter Selection

Types of Experiment
Documenting Experiments

Session 3: Define “Good”.

Evaluation

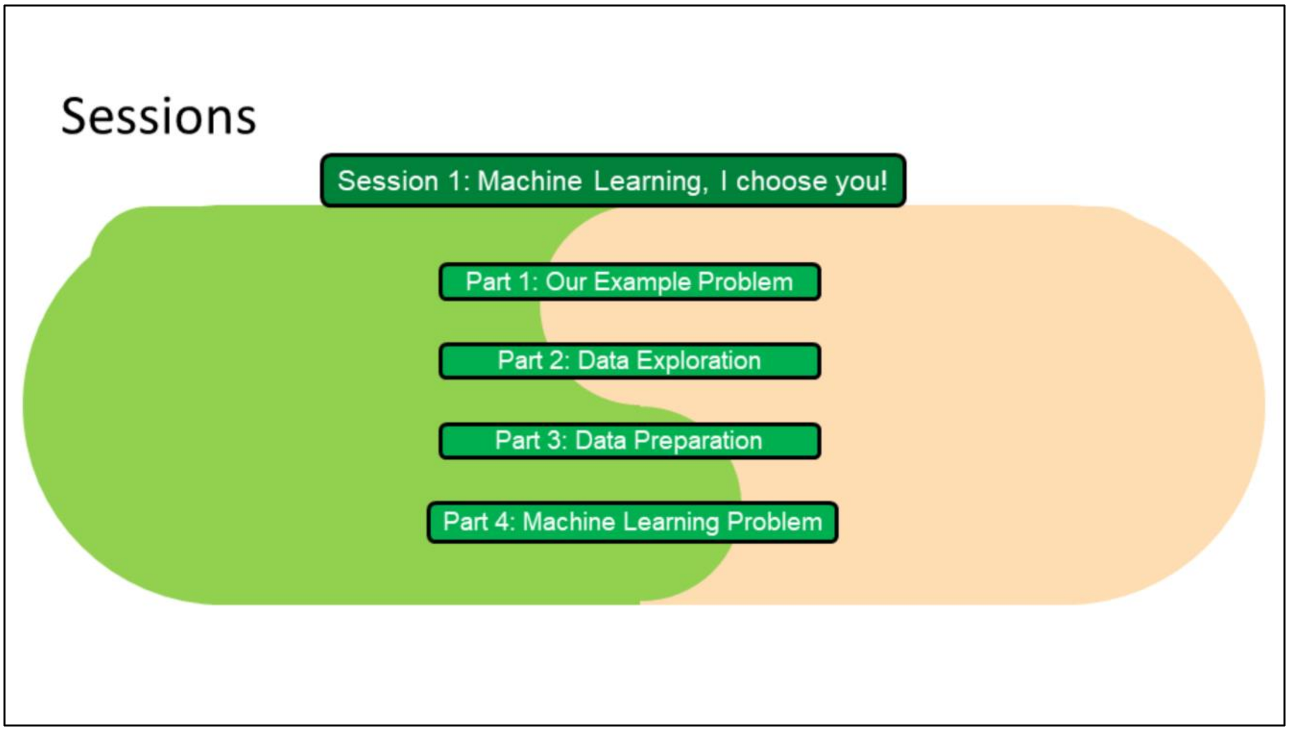
Interaction with other projects

The learning unit is split into three Sessions.

In Session 1 we will lay the groundwork. We will start out with a goal, search for datasets, and explore and process the found dataset. At the end of this session we will have data and a good idea of what machine learning problem we want to solve. On the clean project side, we will focus on how a machine learning project can be set up and how machine learning files interact with standard Python processes.

Session 2 will focus on experimentation. We will select a model and find appropriate hyper-parameters for it. We will do this in three separate experiments, which illustrate three types of experiments you may come across while doing machine learning. On the clean project side, we will talk about how to plan and document experiments.

Session 3 will focus on evaluation of results and utilization of the resulting model. We will talk about how we can evaluate and interpret our model and how we can store and use the result of our learning in other software applications.



Sessions

Session 1: Machine Learning, I choose you!

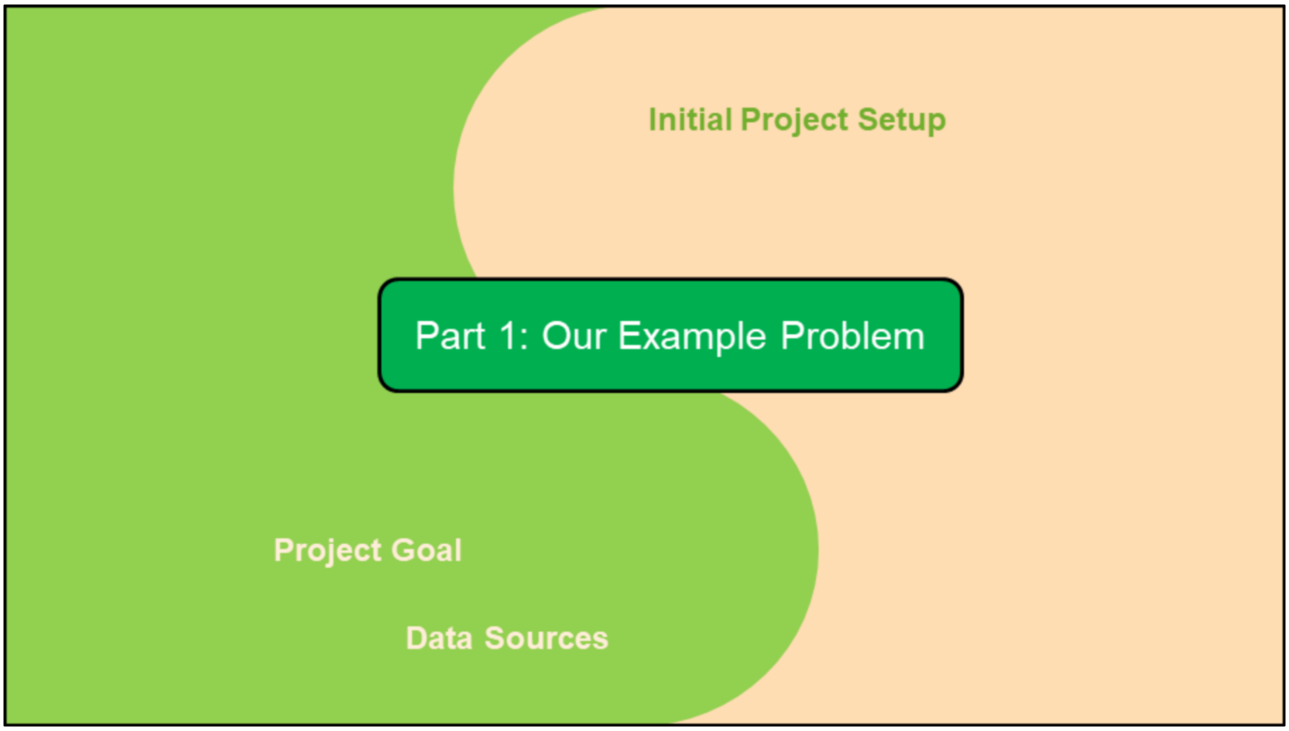
Part 1: Our Example Problem

Part 2: Data Exploration

Part 3: Data Preparation

Part 4: Machine Learning Problem

Today we will discuss the four parts mentioned in the slide.

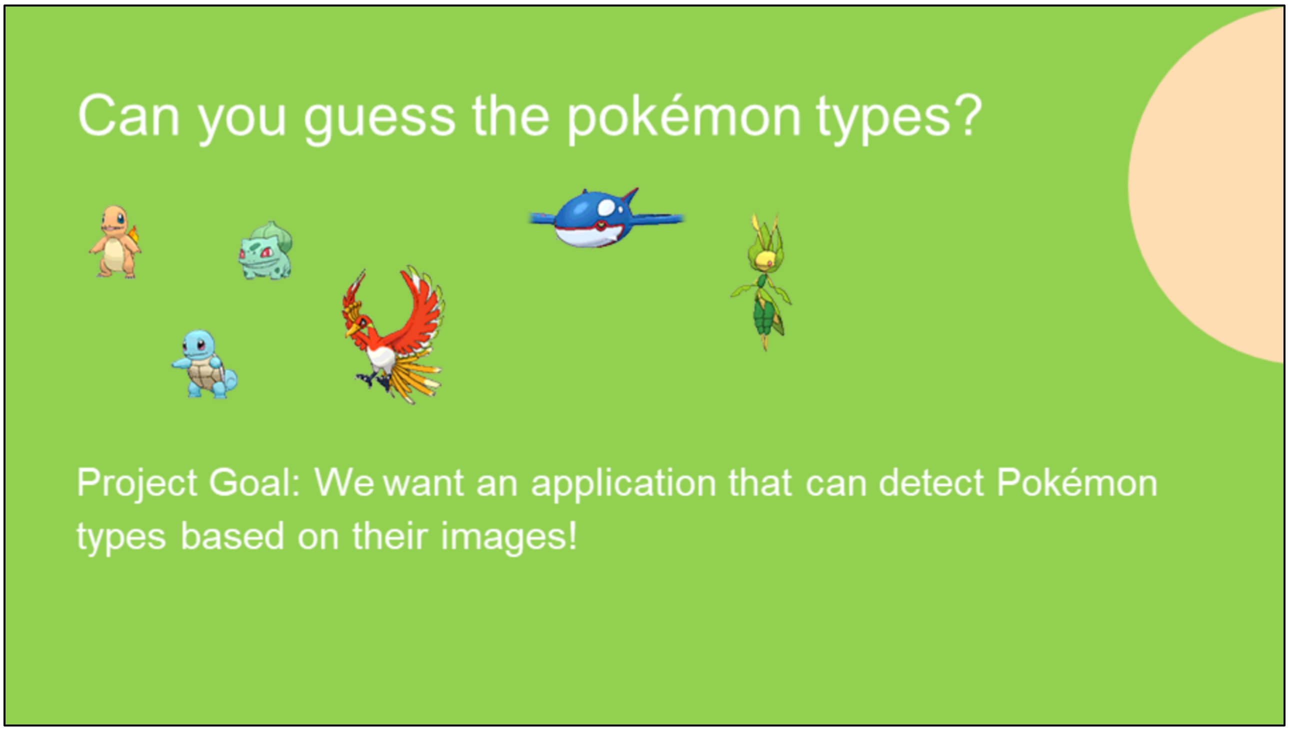


Initial Project Setup

Part 1: Our Example Problem

Project Goal

Data Sources

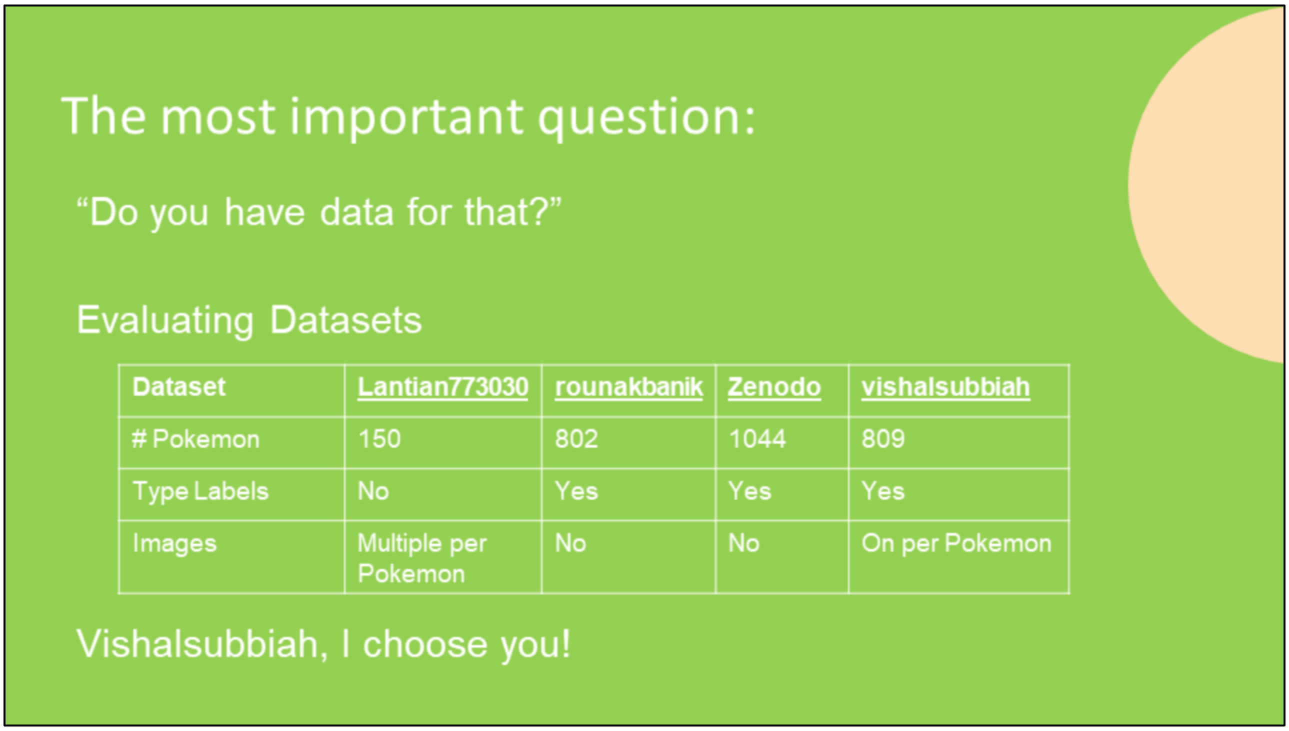


Can you guess the pokémon types?

Project Goal: We want an application that can detect Pokémon types based on their images!

Let's start with our example project. The problem we want to solve is Pokémon classification.

More specifically, we would like to be able to identify the type of a Pokémon based on its image.
Here, we should note that a Pokémon can have more than one type.



The most important question:

"Do you have data for that?"

Evaluating Datasets

Dataset	Lantian773030	rounakbanik	Zenodo	vishalsubbiah
# Pokemon	150	802	1044	809
Type Labels	No	Yes	Yes	Yes
Images	Multiple per Pokemon	No	No	On per Pokemon

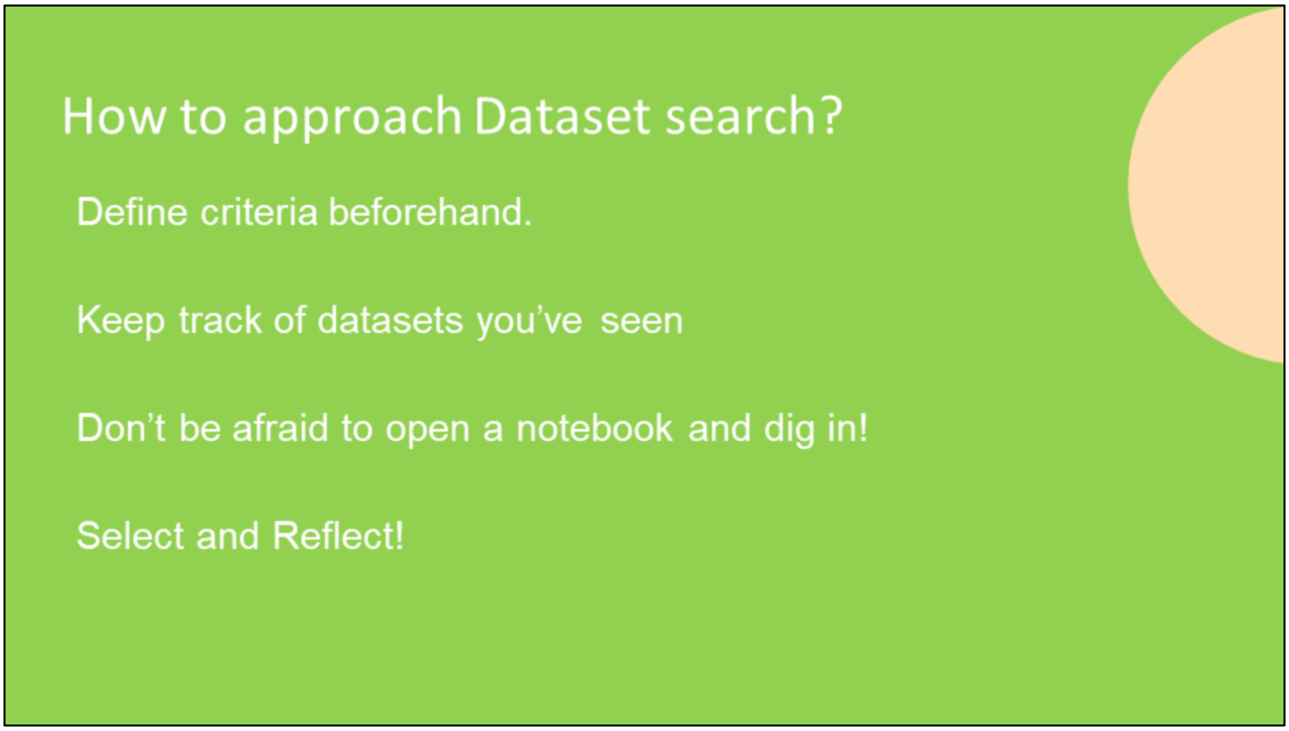
Vishalsubbiah, I choose you!

The first thing you should do when having your goal, is search for datasets. Data is the most crucial part of Machine Learning. As the saying goas: Garbage in -> Garbage out. Without good data, you won't be able to learn a good model.

For our running example, we want a dataset that fulfills the following requirements:
- A high number of Pokémon should be represented in the dataset.
- The dataset should contain information about Pokémon types.
- The dataset should contain images for each Pokémon. The more the better.

We found four different datasets. There are quite a few more, but this is one of the places where take the liberty of not being through because our main goal is to build an example for teaching. Of the four datasets, only one the one from Vishalsubbiah has type labels and images. With this reasoning, we choose this dataset. Unfortunately, it only has one image per Pokémon, which is not optimal.

From the table of collected data, we can also see that…
- At least1044 Pokémon exist, meaning our chosen dataset likely doesn't contain all generations of Pokémon.
- There is a dataset with more images per Pokémon. While it only covers a small number of Pokémon, this may be an option to get more image for training data.



How to approach Dataset search?

Define criteria beforehand.

Keep track of datasets you've seen

Don't be afraid to open a notebook and dig in!

Select and Reflect!

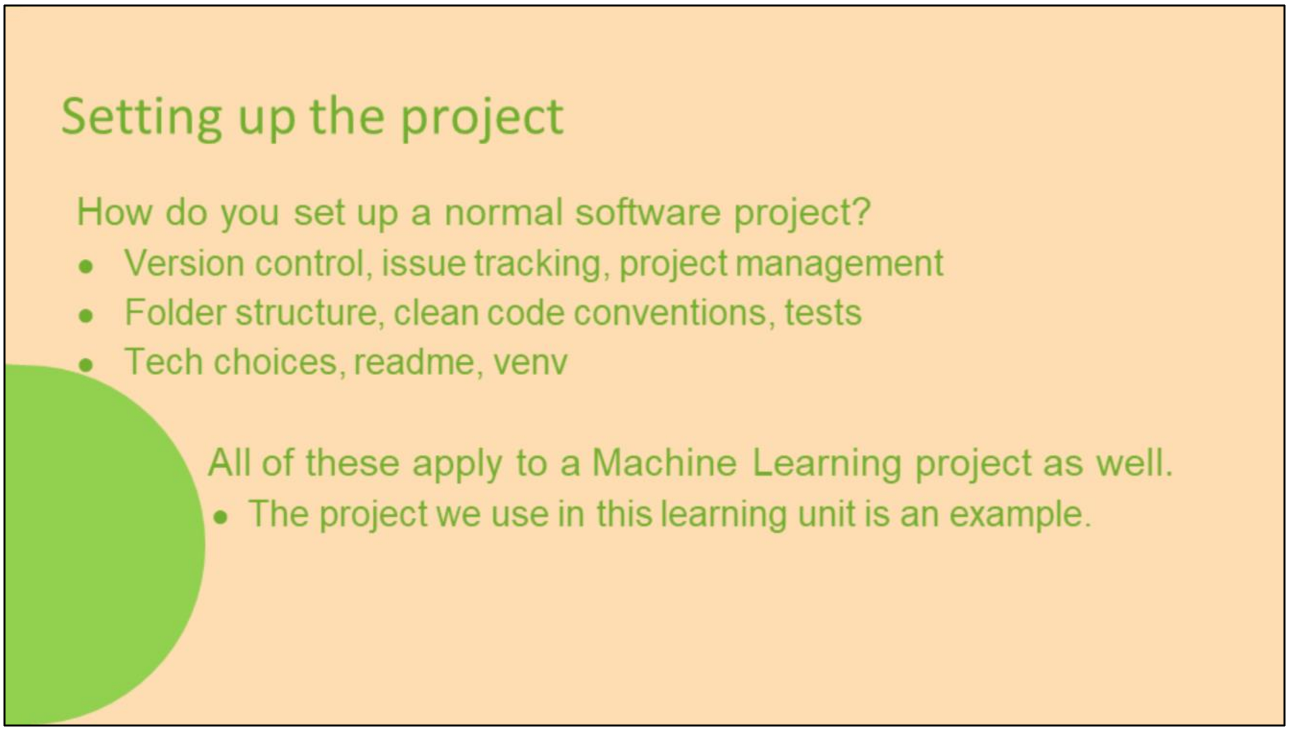
Unless you have a project that is already tied to a data source, you will have to find good data yourself. Even if you have good data sources, you often can complement them with other data.

Before you start the search, you should think about and note down your requirements. These should generally be based on what information should be contained in the dataset and which resolution / quantity it should exhibit. At this stage it is not important for the data to be in the right format. You will do Data analysis and processing anyway.

After you have defined your criteria, it is time to search. It is a good idea to keep track of all datasets you have seen before in a table evaluating your requirements. This gives you an overview later. It can be a good idea to also include datasets that don't fulfill all requirements. On the one hand, this means you can avoid looking at the same dataset twice (more likely to happen if there are a lot of datasets or multiple team members doing the each). On the other hand, you could end up in a situation in which you need to synthesize two datasets and merge them. It's good to know your options early.

Extracting the criteria may require some more detailed analysis. Don't be afraid to already load them into a notebook and play around with them if you can't find all details on their website.

After collecting datasets, you apply your criteria and select the best one / best ones. You should reflect on your choice. Is it perfect? Do you see weaknesses that you need to keep an eye on and are there possible solutions that you could try?



Setting up the project

How do you set up a normal software project?

- Version control, issue tracking, project management
- Folder structure, clean code conventions, tests
- Tech choices, readme, venv

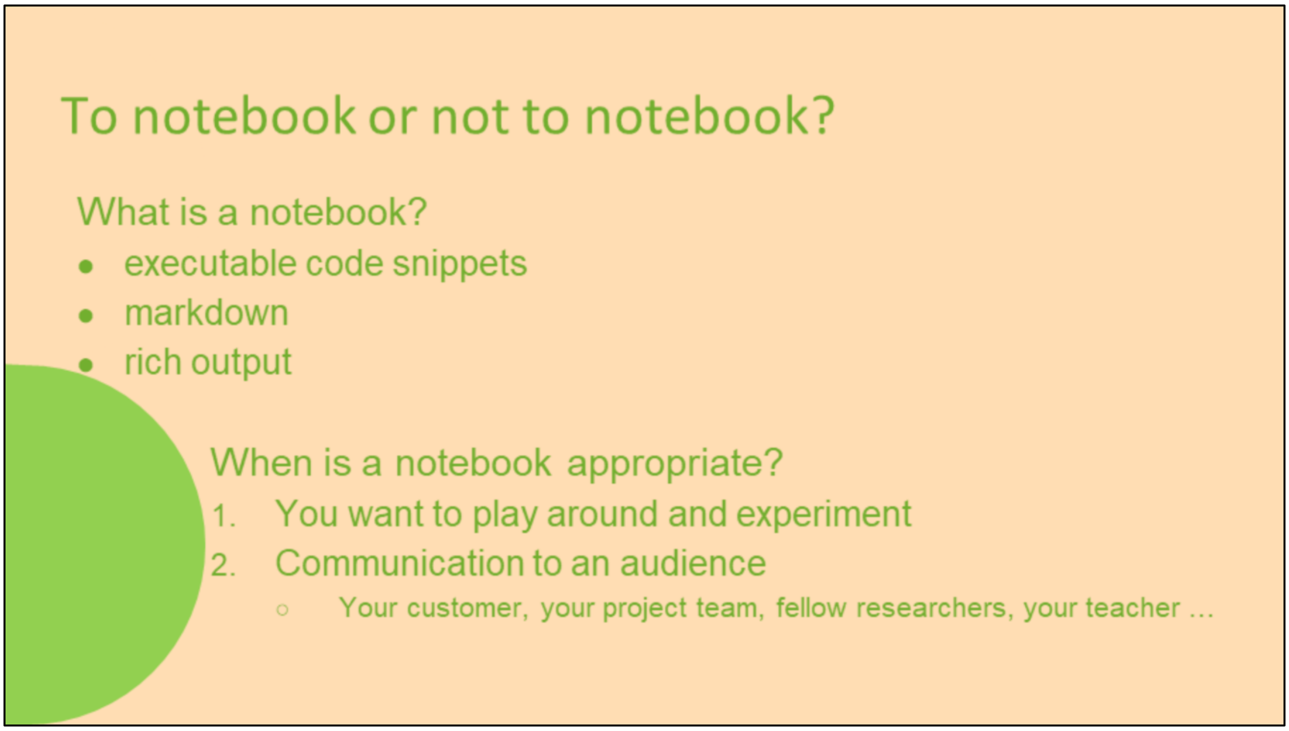
All of these apply to a Machine Learning project as well.

- The project we use in this learning unit is an example.

Once you start your project, you should also set up a source code repository. There are quite a lot of best practices, tools and processes related to a software project. The slide names a few. They also apply to machine learning projects.

This may be a good point to look over the repository of the example project. It showcases one possible setup for a machine learning project. You don’t know all of the content yet, but you can do what you should always do with a project on first glance: look over the Readme file. Try to understand the project scope and it’s folder structure.

Just as with normal Python projects, there are different ideas on how to set up a project in a clean way (e.g., whether code should be in a dedicated source folder). Generally, you should find a good practice for you and your team and stick to it. We will go through more parts of the project as they become relevant to what we are talking about.



To notebook or not to notebook?

What is a notebook?

- executable code snippets
- markdown
- rich output

When is a notebook appropriate?

1. You want to play around and experiment
2. Communication to an audience
   - Your customer, your project team, fellow researchers, your teacher …

One particularity of the Data Science / Machine Learning community is that they love their notebooks. This is not a coincidence. Python notebooks are extremely useful.

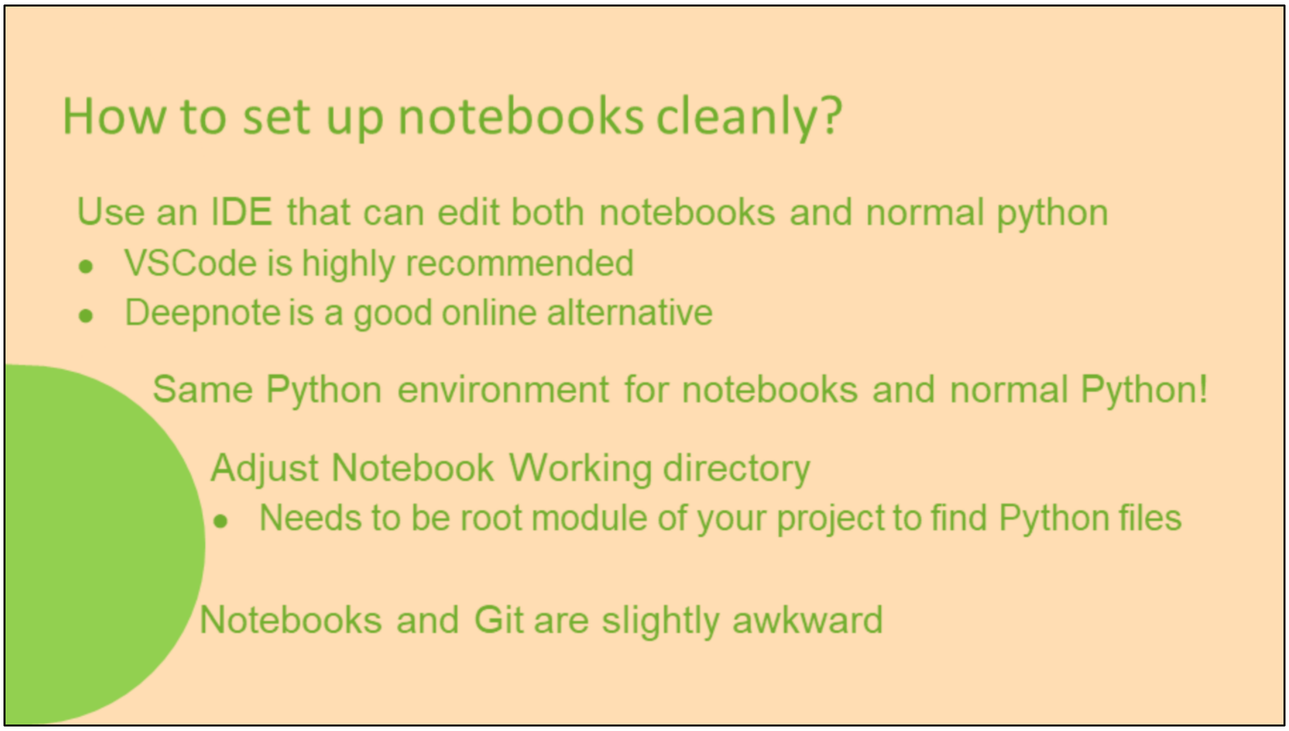
In short, a notebook is an environment with cells containing code that can be executed. This is not dissimilar to the REPL mode of Python, where you enter commands and see the results. However, it is more sophisticated. It can execute whole blocks of code rather than single lines. And it can show complex output, like graphs, instead of just strings. This makes it very useful for data science, where you can get your graphs in the same document as your source code.
Notebooks can also be annotated extensively with Markdown text to write sophisticated texts with tables and figures, if you need to.

When do you use notebooks? There are a few occasions where they shine:

- You can use a notebook as an experimentation environment for yourself. For example, it may make sense to keep a notebook file out of version control to just play around with things before you set them into stone and put them into your more permanent code. This can be a good way to figure things out. However, the resulting notebooks tend to be messy and hard to read / understand so it’s often not a good idea to put them into the project proper.
- You can use notebooks as a communication tool. This is making extensive use of their markup capabilities to essentially write an essay that is complemented with source code. Your audience may vary. Maybe you want to document a decision for your project team. Maybe you want to report a result to your customer. Or maybe

you want to document an experiment for a research paper. Or maybe you want to teach a machine learning method to someone.

This last point is why so many online tutorials rely on notebooks. It’s an easy communication tool for teaching. It’s also why the example project contains a few more notebooks than would maybe be necessary: they are a better way to communicate certain information to you than documented source code or slides.



How to set up notebooks cleanly?

Use an IDE that can edit both notebooks and normal python
- VSCode is highly recommended
- Deepnote is a good online alternative

Same Python environment for notebooks and normal Python!

Adjust Notebook Working directory
- Needs to be root module of your project to find Python files

Notebooks and Git are slightly awkward

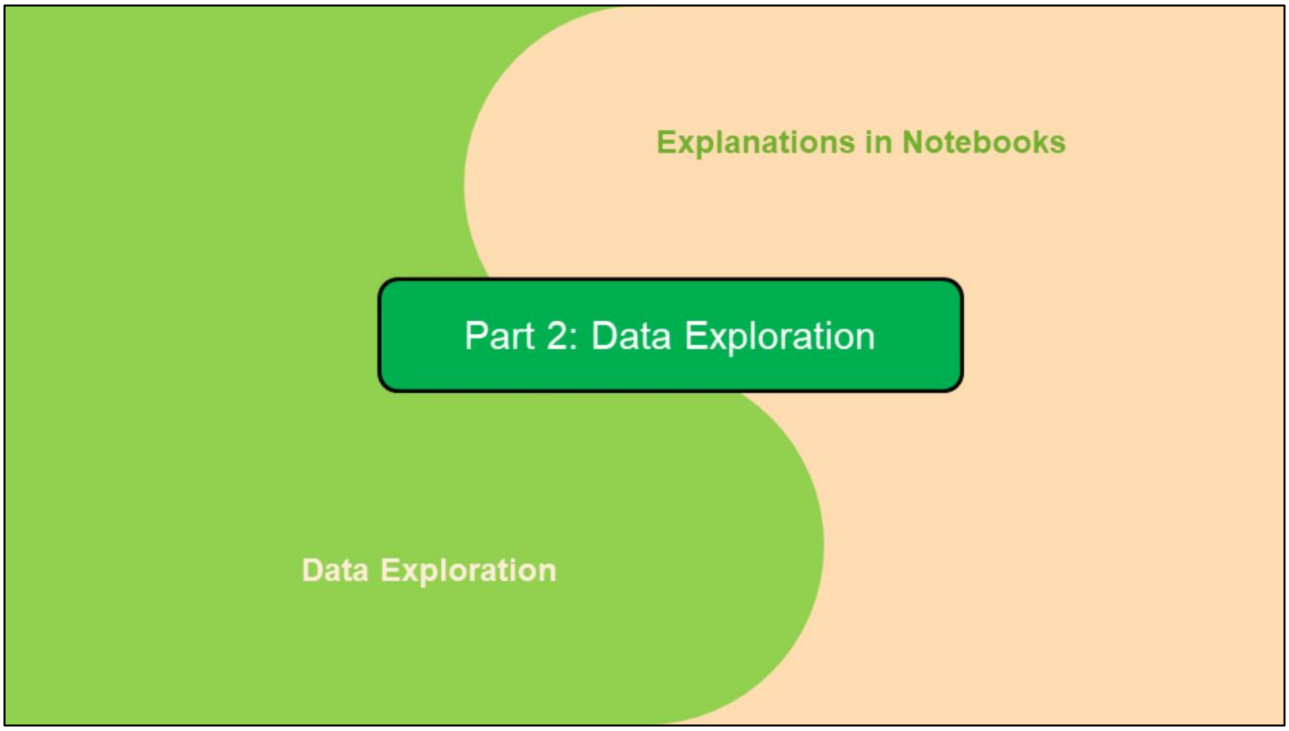
Setting up notebooks cleanly can be a bit tricky. Generally, what I recommend is to set up an environment in which you can easily edit and run traditional Python files as well as notebooks. This way you get a coherent programming environment for your whole project.

One IDE that does this well is VSCode. It has plugins for Python files and Jupiter notebook and can be made to use the same Python environment for both. Essentially, you can create a virtual environment, install all of your libraries in it and then use it for both your normal python and your notebook executions and always be certain to use the correct versions of your libraries.

Deepnote is a good online alternative to VSCode. This is an online environment that lets you edit and execute Python files and Jupiter notebooks. It comes with a generous teaching plan that gives you access to a cloud server that should be sufficient for small-scale machine learning projects.

When you use notebooks, you may have to do some extra work to be able to import your Python files. The default working directory of a notebook is the folder it is placed in. If your notebook is in your project root folder, this should work well. If have decided to have a different structure – maybe a dedicated notebook folder, like our example project, then you will have to work a bit. You can find example code at the beginning of each of our projects that automatically sets the working directory to the correct folder.

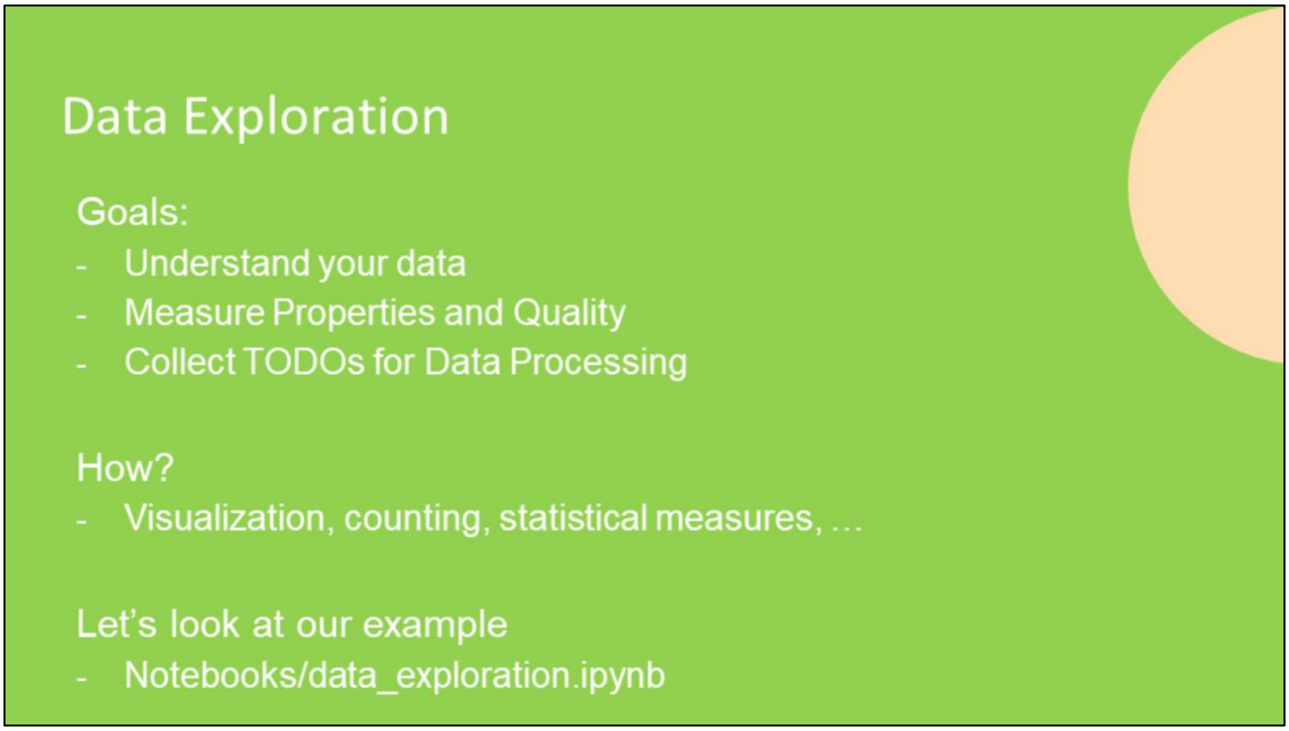
Notebooks also are compatible with version control now (this used to be somewhat of an issue). You can do all of your usual git operations on notebooks. The only thing you need to be aware of is that the output is also in version control. This means if you rerun the notebook it will think there is an update that could be committed. It’s easy to clutter your git repository by committing the same output of running the same cells at different times. Be mindful of that and only commit notebooks if they have intentional changes.



Explanations in Notebooks

Part 2: Data Exploration

Data Exploration



Data Exploration

Goals:
- Understand your data
- Measure Properties and Quality
- Collect TODOs for Data Processing

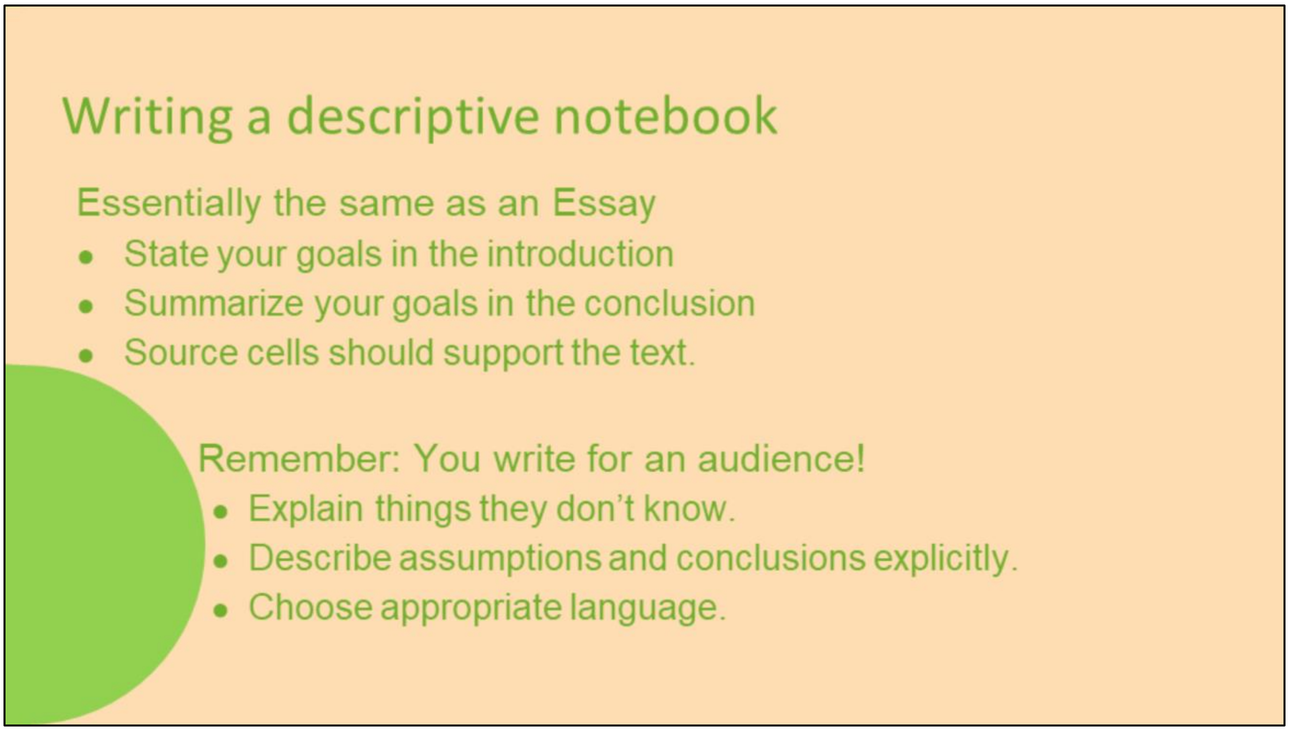
How?
- Visualization, counting, statistical measures, …

Let’s look at our example
- Notebooks/data_exploration.ipynb

After selecting your dataset, the next step is exploring it. Your goal here is to understand your data and its structure, properties and quality. This will give you some points that will need addressing during data processing or that may have an influence on the machine learning.

The “How” depends on your data. This is essentially Data Science and will depend a lot on which domain you are working with.

In our example, we have images and a very simple tabular dataset. To see how we explored this dataset, please refer to *data_exploration.ipynb.*



Writing a descriptive notebook

Essentially the same as an Essay

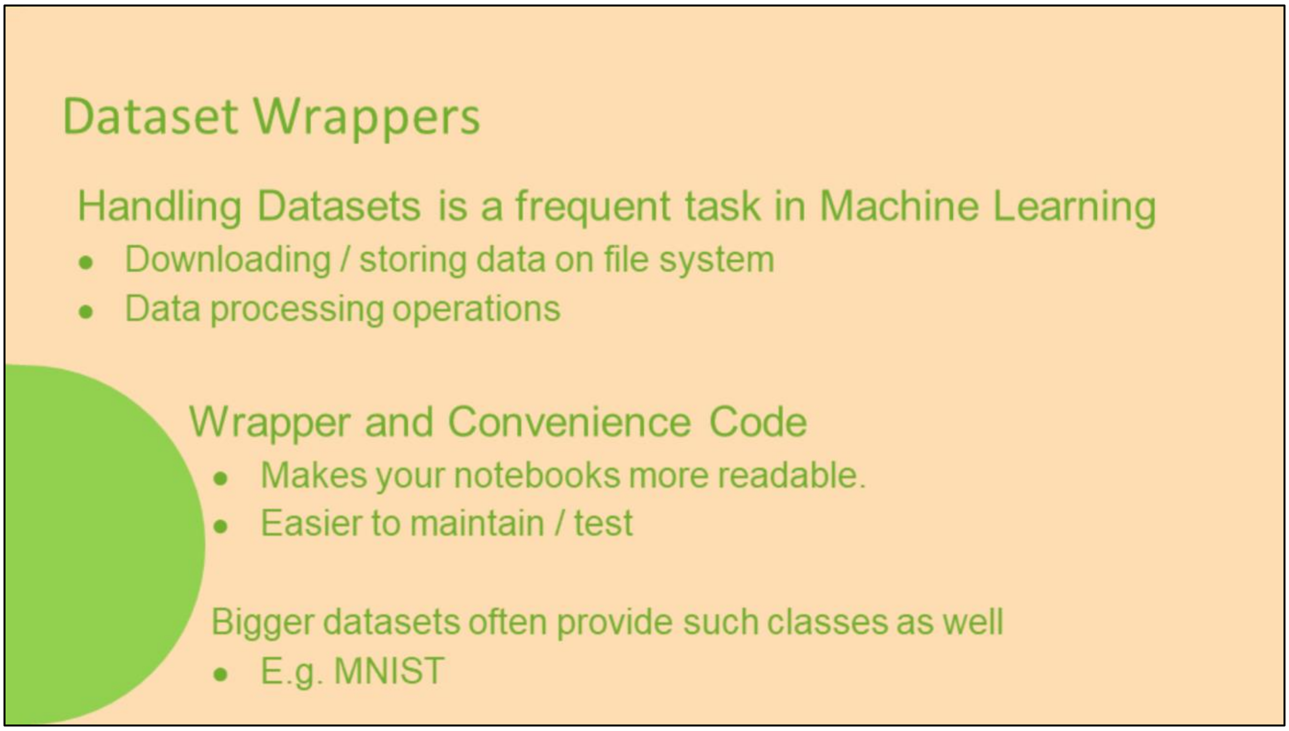
- State your goals in the introduction
- Summarize your goals in the conclusion
- Source cells should support the text.

Remember: You write for an audience!

- Explain things they don't know.
- Describe assumptions and conclusions explicitly.
- Choose appropriate language.

Now that we've looked into our first notebook, let's talk a bit about how to write a good notebook.

This refers to the second use case for a notebook, where you try to communicate something to an audience. Essentially, you should regard your notebook as an essay or text. You should state the purpose and goals of the notebook clearly in the beginning and circle back to them by the end. The source code in this type notebook is there to support your text – by producing results or providing mini-experiments the reader can carry out. Remember that you write for an audience. Make sure that you're writing in a way they can understand.



Dataset Wrappers

Handling Datasets is a frequent task in Machine Learning

- Downloading / storing data on file system
- Data processing operations

Wrapper and Convenience Code

- Makes your notebooks more readable.
- Easier to maintain / test

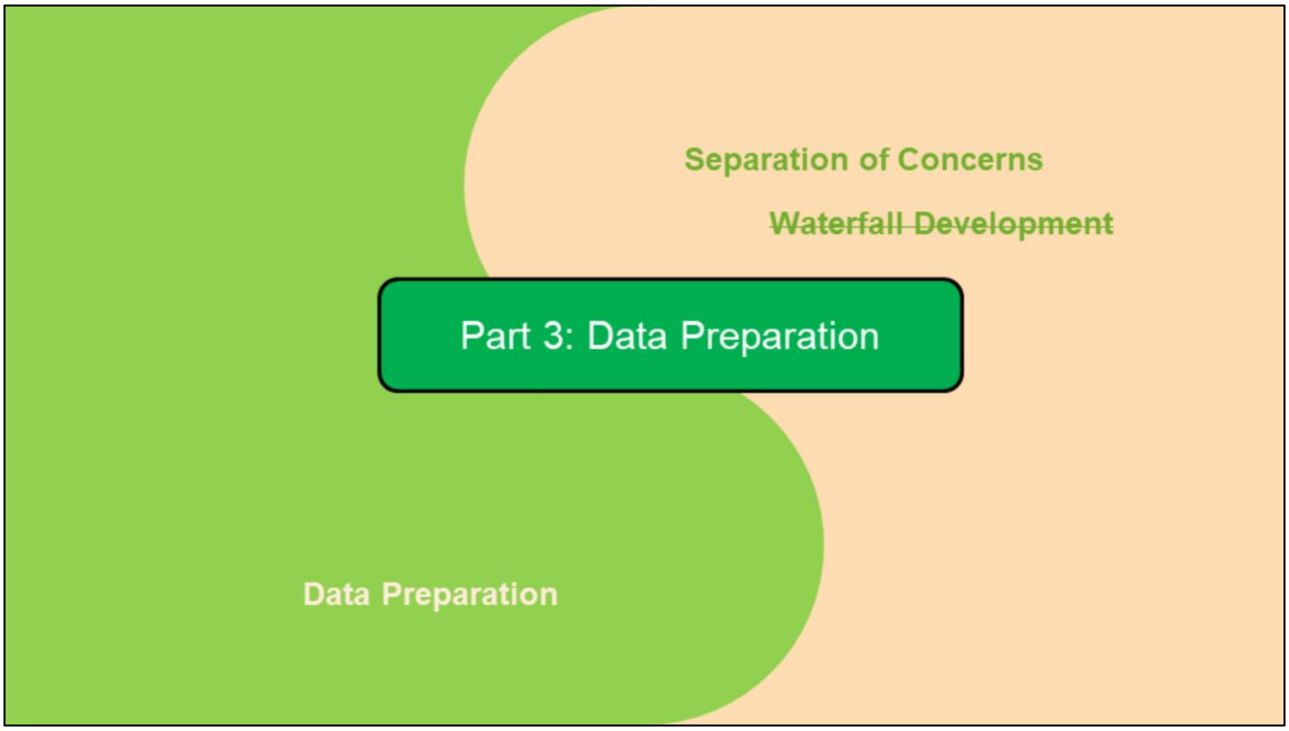
Bigger datasets often provide such classes as well

- E.g. MNIST

One thing that you will have seen in the notebook is that the example project contains a class *Dataset* that handles download and storage of data. This is a good convention to follow in a machine learning project. You will likely accumulate quite some code related to downloading data, restructuring it, analysing it etc. It usually is a good idea to modularize this code into Python files to remove some clutter from your notebooks. This also has the added benefit of not having to repeat this code if you have multiple notebooks.

Our *Dataset* class in particular abstracts from the location of our data. It knows where it can download the dataset and will do the first time it is requested. Afterwards it will be stored in the project folder for quicker access. From point of view of our notebook, this was transparent. We just created the Dataset and asked it for data. A similar functionality will be available for the prepared data we will create in a alter step.

This is a best practice that you may also encounter if you use some of the more well-maintained datasets, such as the MNIST dataset. This dataset is imported as a module and works similarly by providing the data via a Python interface that takes care of downloading and storing data.

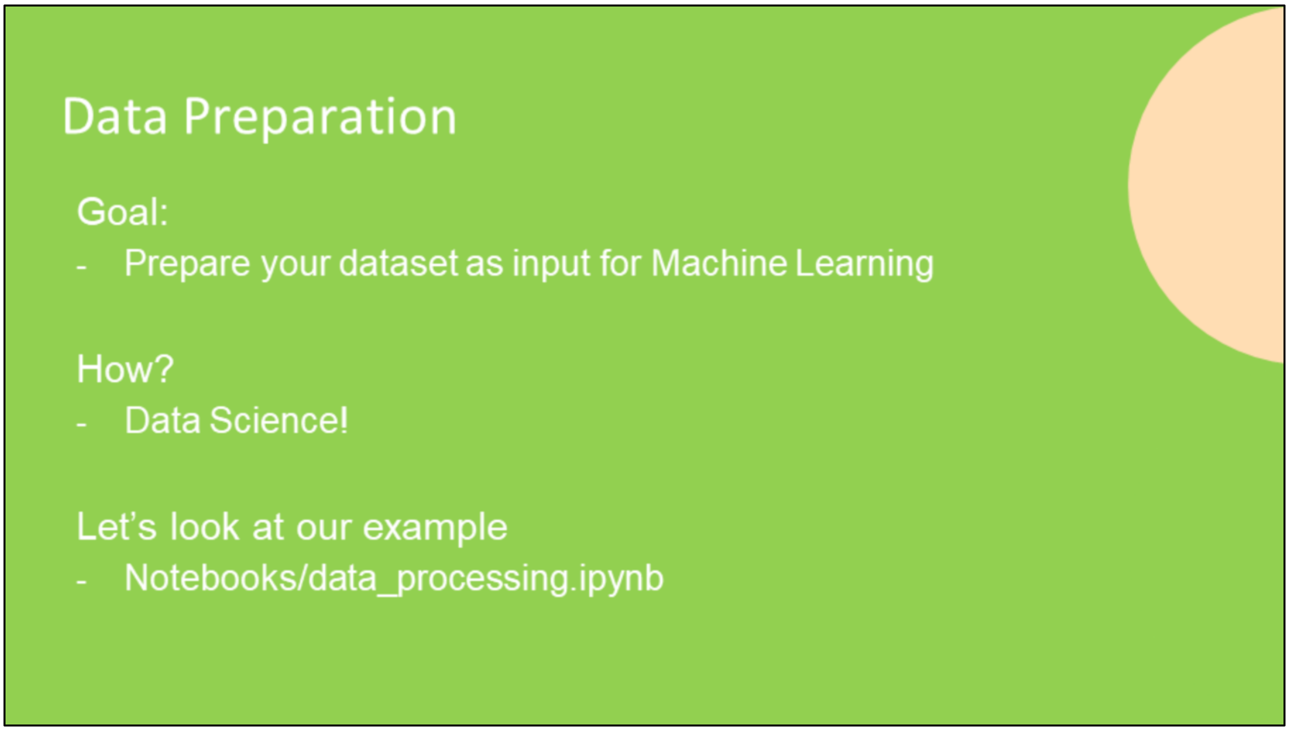


Separation of Concerns

~~Waterfall Development~~

Part 3: Data Preparation

Data Preparation



Data Preparation

Goal:
- Prepare your dataset as input for Machine Learning

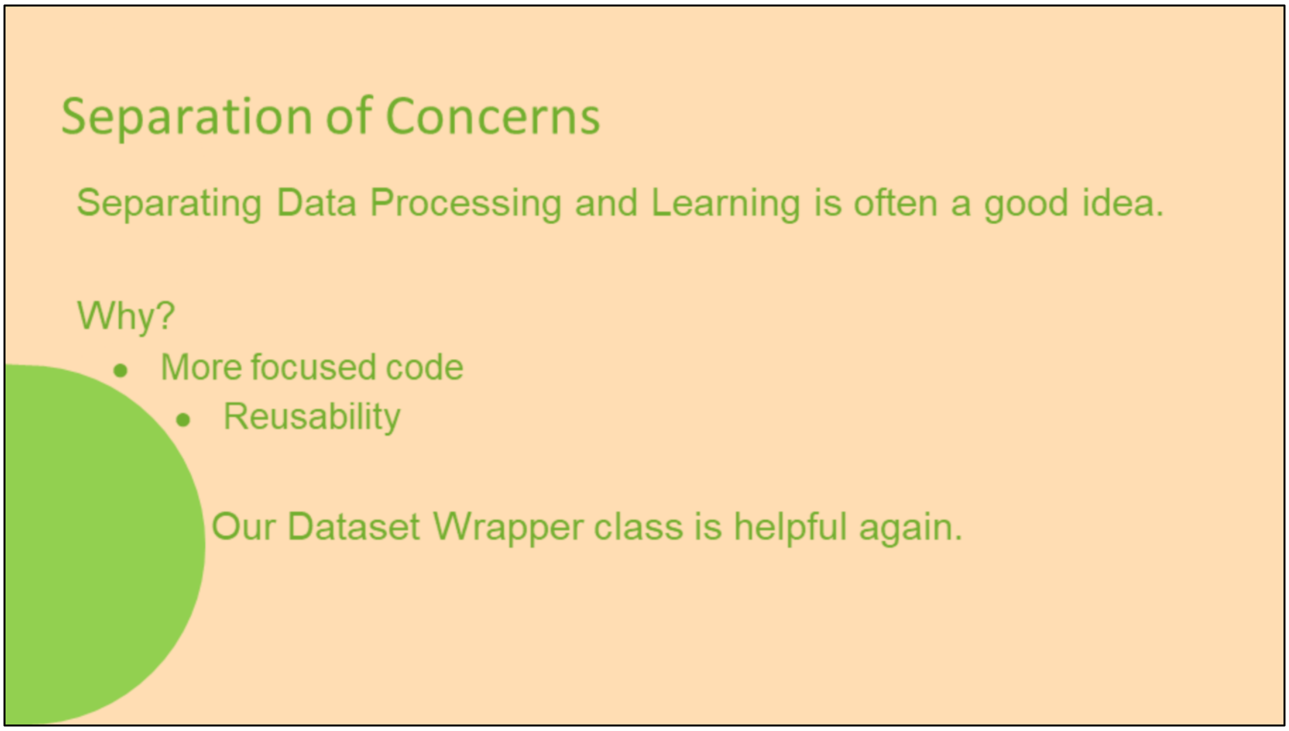
How?
- Data Science!

Let's look at our example
- Notebooks/data_processing.ipynb

After exploring our dataset, it is time for the next step: Data preparation.

During this step we prepare our data to create the direct input of the machine learning model. In the case of our supervised example, we want to create the X dataset containing all features and the y dataset, containing labels.

How is this done? It's data science again. Again, the actual methods strongly depend on your problem. An example for our Pokemon classification can be seen in *data_processing.ipynb*.



Separation of Concerns

Separating Data Processing and Learning is often a good idea.

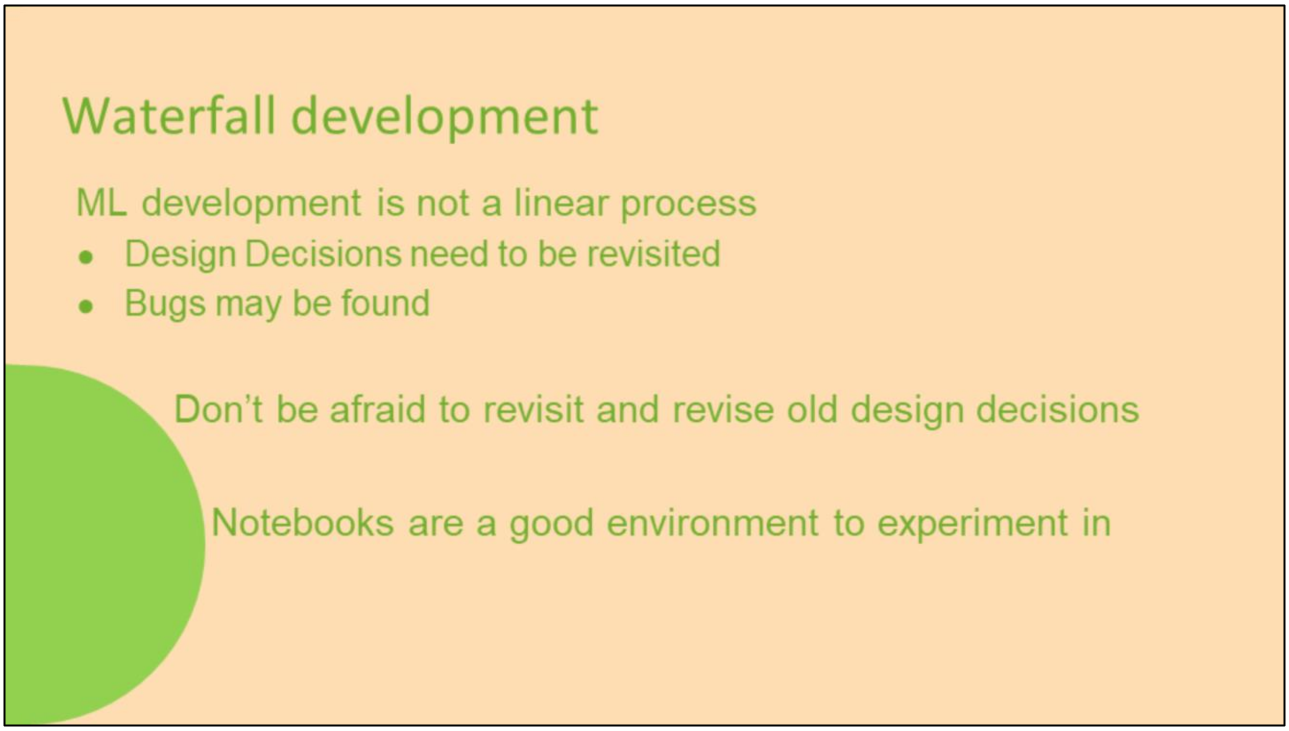
Why?

- More focused code
- Reusability

Our Dataset Wrapper class is helpful again.

On a methodological side, it is often a good idea to separate your code for data processing and learning into different files (or at least notebooks).
As you will see later, learning may be done multiple times for different experiments. Preparing data once and then storing it for easy access can declutter your experiments a lot.

As indicated earlier, our Dataset class is helpful yet again as it also manages storing and loading the prepared data.



Waterfall development

ML development is not a linear process

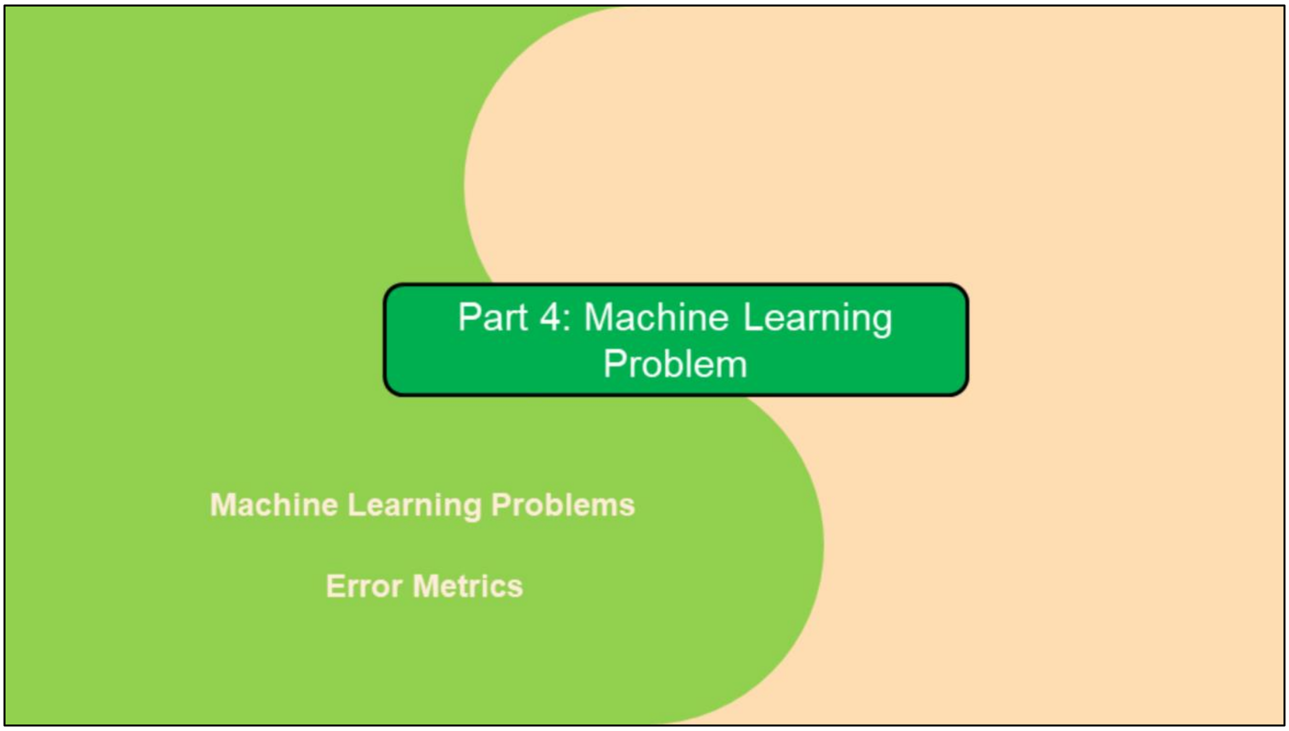
- Design Decisions need to be revisited
- Bugs may be found

Don't be afraid to revisit and revise old design decisions

Notebooks are a good environment to experiment in

One more sidenote is that Machine Learning is not always a linear process. These slides may make it seem like data set selection, data exploration and data preparation are consecutive steps. In reality, they are often more iterative. For example, you may explore a dataset as part of the selection process.

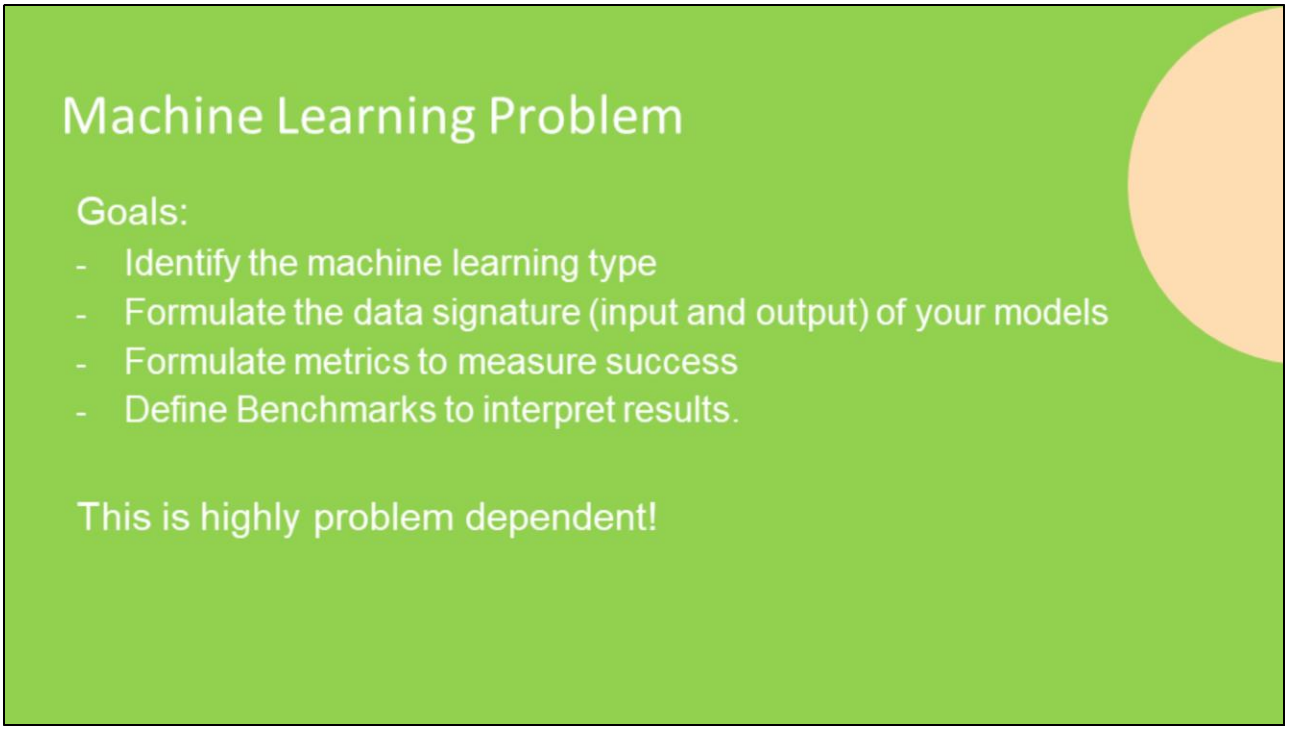
There is a good example of this in our running example. During model selection, we noted that neural networks perform significantly better if we invert the image colors beforehand. After noting this, we went back and made this a part of the data processing notebook.



Part 4: Machine Learning Problem

Machine Learning Problems

Error Metrics



Machine Learning Problem

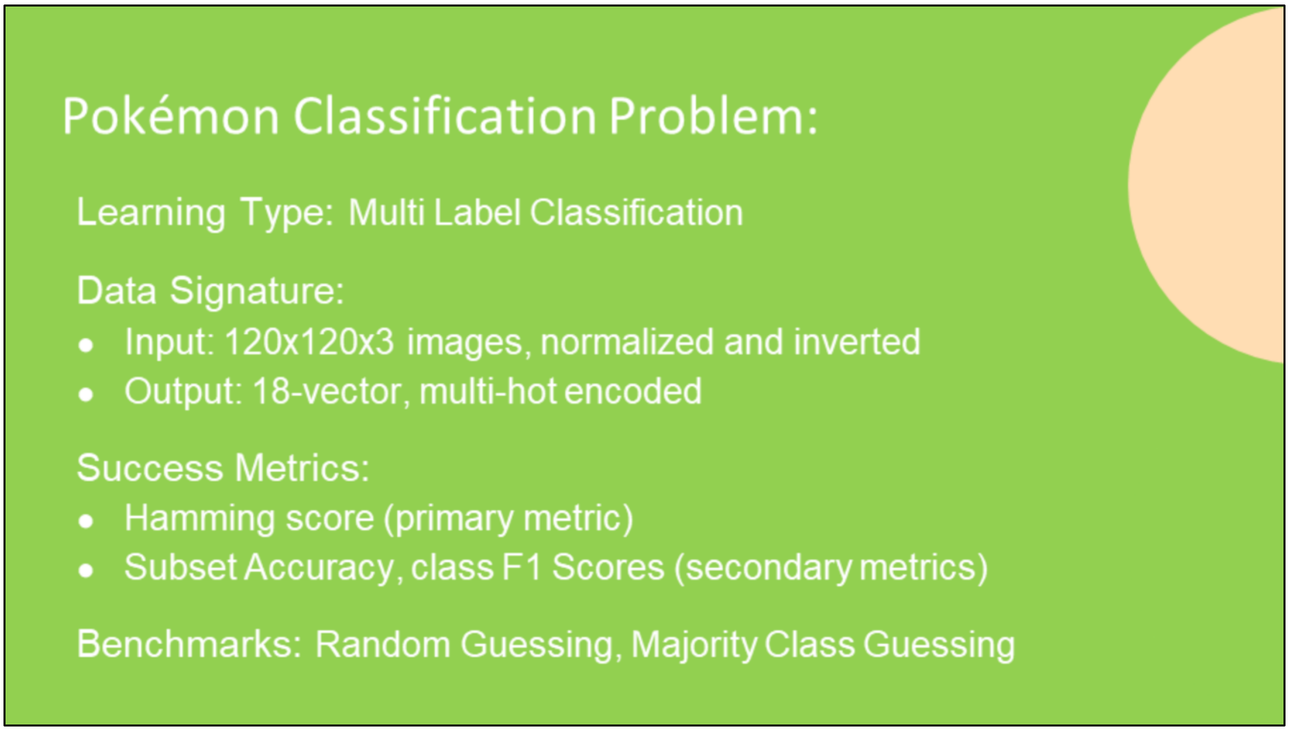
Goals:
- Identify the machine learning type
- Formulate the data signature (input and output) of your models
- Formulate metrics to measure success
- Define Benchmarks to interpret results.

This is highly problem dependent!

Now that we have our data to learn from, it's time to define the machine learning problem. There are a few things you should decide on now:

- Type of machine learning: Is it supervised or unsupervised? Regression or classification? This will likely be obvious from the project goal.
- Data Signature: Which data do you have as input to your model? What type of output do you expect?
- Success Metrics: Which metric will you use to measure your model?
- Benchmarks: What is your benchmark for success? Do you have some simple method / model that you want to beat? Is there a certain value of your success metric that you really want to reach?

It is a good idea to write these down formally before you continue with finding a machine learning model. It will give you an impartial measure of your goals unbiased by the experience of seeing your models in action.



Pokémon Classification Problem:

Learning Type: Multi Label Classification

Data Signature:

- Input: 120x120x3 images, normalized and inverted
- Output: 18-vector, multi-hot encoded

Success Metrics:

- Hamming score (primary metric)
- Subset Accuracy, class F1 Scores (secondary metrics)

Benchmarks: Random Guessing, Majority Class Guessing

Let’s apply this to our running example:

The learning type is a Multi-Label Classification. We have a classification problem where each Pokémon can have one or more types. The data signature of this problem contains images of dimension (120,120,3) as input and a vector of class predictions as output. There are 18 Pokémon types, so the output vector has a length of 18.

Several success metrics can be chosen for a multi-label classification:

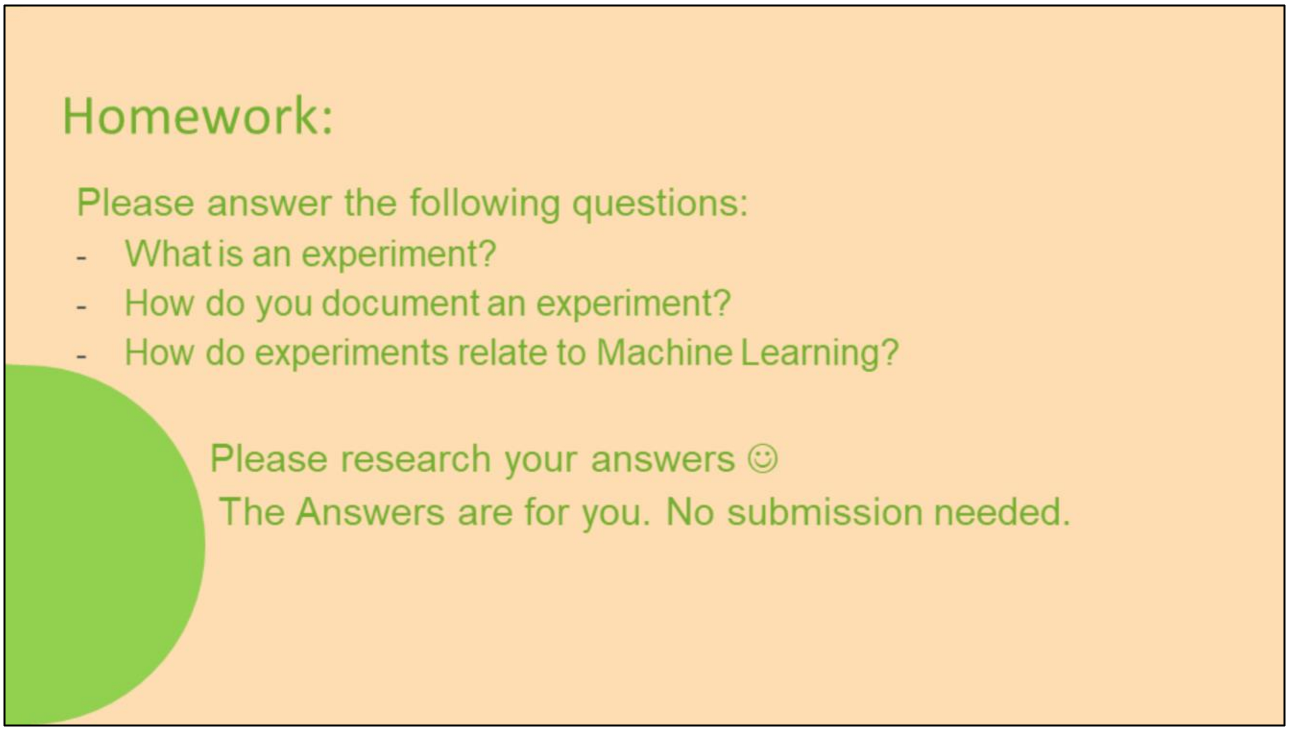
- Subset Accuracy represents the percentage of Pokémon have been guessed correctly. This metric is fairly harsh – it does not reward parial or close guesses.
- The Hamming Score is a more lenient metric. It counts the relation between correct guesses and overall guesses and classes per sample.
- The F1 score is defined for binary classification. We can extend it to the multi-label classification problem by applying it to each individual label and then averaging.

We have selected the Hamming Score as our primary metric. We will use this during experiments to detect which models perform better. Subset Accuracy ad F1 scores will be checked when evaluating the final model. The F1 score will not be averaged. Instead, we will apply it to each class to get a feeling for how well that class is predicted.

There are two natural benchmarks that we want to beat: Guessing Randomly and

always guessing the class with the highest occurrence in our data. If we beat these two benchmarks, we can claim to have learned some meaningful patterns. However, being more meaningful than random with 18 classes is not a highly accurate result. We hope to beat these. scores by a good margin.
A very good result would be close to 1 in hamming score. Due to the small size of the data set we do not think this is realistic. We would be happy with a hamming score of between 0.5 and 0.8.

Homework



Homework:

Please answer the following questions:

- What is an experiment?
- How do you document an experiment?
- How do experiments relate to Machine Learning?

Please research your answers ☺

The Answers are for you. No submission needed.

Questions?