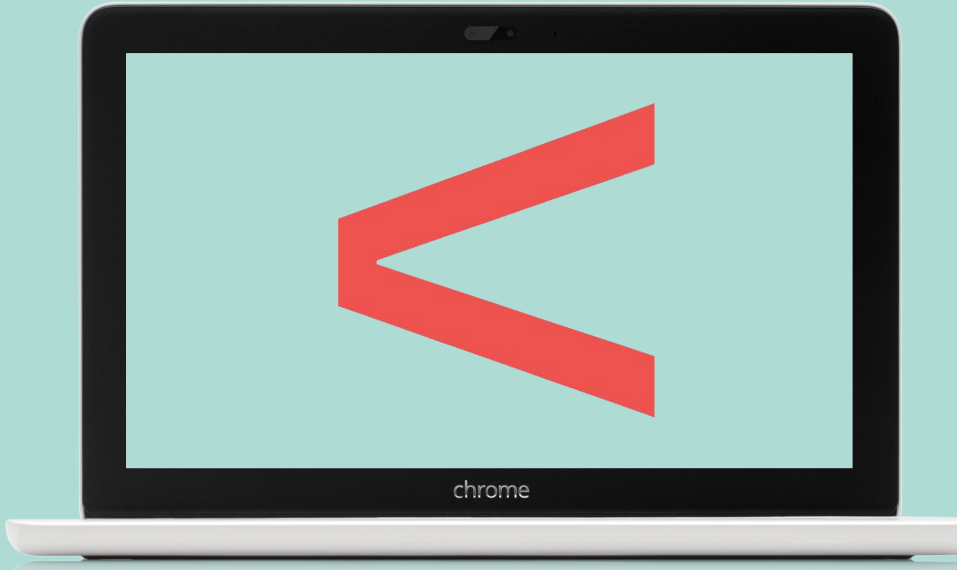
ML Steps



Introduction

The following slides can be considered a high-level guideline to the general steps used in a Machine Learning Pipeline.

There are links embedded in the slides which can be used for further reference and reading. Don’t worry if you don’t do every element of each step, many are optional or context dependent.

Step 1
Collect data

What to be aware of

As you know, ML algorithms learn from the data that you give them. It is important to collect reliable data so that your model can find the correct patterns. The quality of the data that you feed it will determine how accurate your model is.

Make sure you use data from a reliable source, as it will directly affect the outcome of your model. Good data is:

- Relevant
- Contains very few missing and repeated values
- Has a good representation of the various subcategories/classes present.

Try reputable sources such as Statista or government websites. Note that you can access Statista for free with a library card.

Step 2
Prepare the data

What to be aware of

After you have your data, you have to prepare it. This step (EDA) is extremely important to getting a good quality result. As the adage goes: Garbage in, garbage out. You can do this by :

- **Cleaning** the data to remove unwanted data, missing values, rows, and columns, duplicate values, data type conversion, etc. You might have to restructure the dataset or create new features from existing ones (feature engineering).

- **Visualising** the data to understand how it is structured and understand the relationship between various variables and classes present. Correlated features should be considered at this point, though in general we can try the model with all features first and only start removing them after an initial run. Any outliers should be identified (though not necessarily removed), skewness & kurtosis can be investigated etc.

- **Randomising** your data. This helps make sure that data is evenly distributed, and the ordering does not affect the learning process. Consider **class imbalance** at this stage for classification models, such as by using Stratified K-Fold. SMOTE and other techniques such as under- and over- sampling may also be considered.

What to be aware of

Preparation, continued:

- **Splitting** the cleaned data into two sets - a training set and a testing set. The training set is the set your model learns from. A testing set is used to check the accuracy of your model after training. The rule of thumb here is to have between a 70/30 and 80/20 split. With large enough datasets, it is also possible to have a validation hold out set. We also split into X and y for both, with X being the features and y being the target.

- **Feature scaling**. This must only be done AFTER splitting in order to prevent **data leakage**. This step largely depends on the type of algorithm you are using and the data distribution, but in general it can be split into **Normalisation** and **Standardisation**.

  Normalisation is highly affected by outliers. Standardisation is slightly affected by outliers. Normalisation is considered when the algorithms do not make assumptions about the data distribution. Standardisation is used when algorithms make assumptions about the data distribution. [1]

What to be aware of

Preparation, continued:

We can also look at **dimension reduction** techniques at this stage (removing features).

One such technique is called **backwards elimination**.

The goal is to identify a subset of the most relevant and informative features in the dataset that can be used to train the model. This is done by starting with all of the available features and iteratively removing the least significant ones until a desired subset is obtained.

Backwards elimination can be a useful technique for reducing the complexity of a machine learning model and improving its interpretability. It can also help to reduce overfitting, which occurs when a model is too complex and has too many parameters relative to the size of the training data, leading to poor generalization to new data.

Another method is **Principal Component Analysis** (**PCA**) but that is outside the scope of this course.

See “The curse of dimensionality” for more information.

Step 3
Choose a model

What to be aware of

A machine learning model determines the output you get after running a machine learning algorithm on the collected data. It is important to choose a model which is relevant to the task at hand.

There is no one perfect model; all have their advantages and drawbacks. Which one is suited to your problem will require some reading and experimentation. See the "No Free Lunch" theorem for more information.

Apart from this, you also have to see if your model is suited for numerical or categorical data and choose accordingly. In our first project, we are considering a classification problem. Here you can see a (non-exhaustive) list of some classification algorithms.

To learn more about classification models, read this article and this other article.

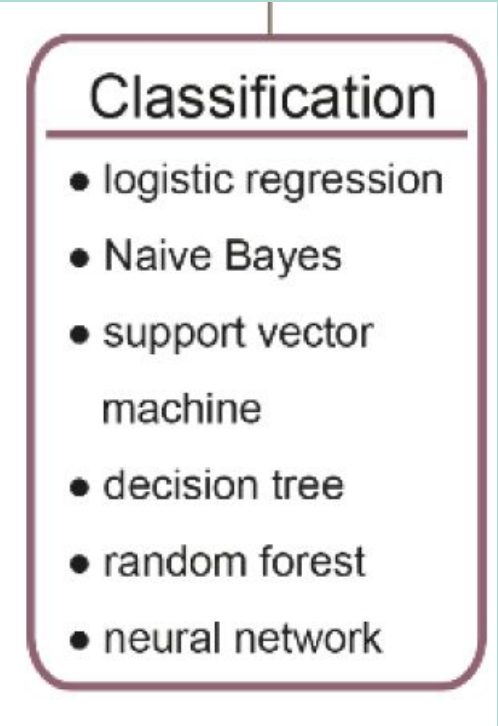


Image source: https://mdpi.com

Step 4
Train the model

What to be aware of

In training, you pass the prepared data to your machine learning model to find patterns and make predictions. It results in the model learning from the data so that it can accomplish the task set.

Assuming that you have already split and scaled the data appropriately, you can use the sklearn module to load in your algorithm of choice and fit it to the training data which you have prepared.

At this stage we can leave default hyperparameters and tune them later, after the evaluation stage.

Step 5
Evaluate the model

What to be aware of

After training your model, you have to check to see how it's performing.

This is done by testing the performance of the model on previously unseen data. The unseen data used is the testing set that you split the data into earlier.

If testing was done on the same data which is used for training, you would not get an accurate measure, as the model is already used to the data, and finds the same patterns in it, as it previously did. This will give you disproportionately high accuracy.

When used on testing data, you get an accurate measure of how your model will perform and its speed.

You can refer to the Evaluation Metrics slides for more information.

Step 6
Tune hyperparameters

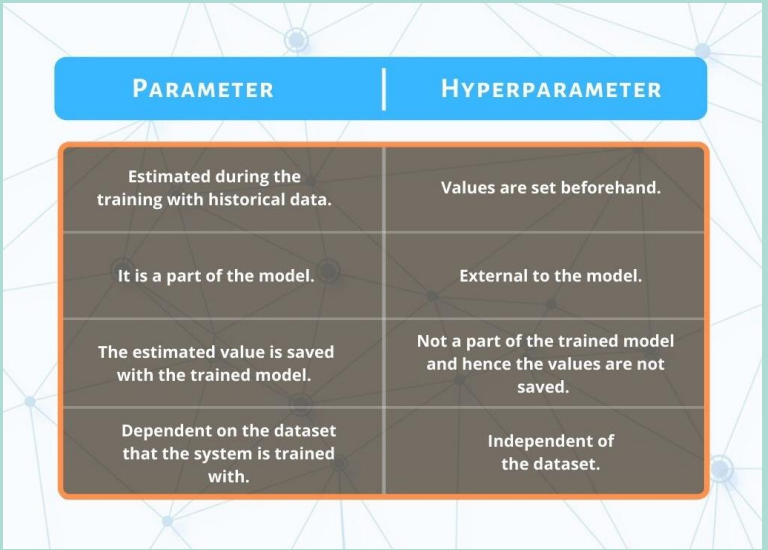
What to be aware of

Once you have created and evaluated your model, see if its accuracy can be improved in any way. This is done by tuning the hyperparameters present in your model.

Hyperparameters are the variables in the model that you can specify. At a particular value of your hyperparameter, the accuracy will be the maximum. Hyperparameter tuning refers to finding these values.

After model evaluation, we will have a better idea of what values we might need to tune. After that, we can use techniques built into scikit-learn to optimise further.

Two of the easiest methods for doing this are **Grid Search** and **Random Search**.



PARAMETER	HYPERPARAMETER
Estimated during the training with historical data.	Values are set beforehand.
It is a part of the model.	External to the model.
The estimated value is saved with the trained model.	Not a part of the trained model and hence the values are not saved.
Dependent on the dataset that the system is trained with.	Independent of the dataset.

Image source: https://www.hitechnectar.com/blogs/hyperparameter-vs-parameter/

Step 7
Iteratively improve

What to be aware of

At this stage, based on what we learned in the training, evaluation (and tuning) steps, we will make small changes to the model and reassess its performance.

This is a very iterative process and we may need to do this many times before settling on a satisfactory solution to our problem.

Step 8
Deploy the model

What to be aware of

In the end, you can use your model on unseen data to make predictions accurately.

This is one of the most satisfying parts of the entire pipeline. Unfortunately, as data scientists or engineers, we may not often get the chance to do it.

However this does not mean we shouldn't learn how! For Python, a common method for deploying a ML model is using Flask.

It is also possible to connect directly to a Github repository with something like Streamlit Cloud.

Conclusion

As you can see, there are many steps and many possible directions you can go when preparing a machine learning model.

It can seem daunting at first, but once you break it down, each step is manageable.

Often, the best approach is to generate the baseline performances of different algorithms with your dataset before doing any of the performance enhancements we describe above.

This will give you a good platform to work from and you can more quickly identify the most promising algorithms for your problem.

Good luck!