Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

In-Line with Shai's Machine learning engineering internship, chapter 1 "The Machine Learning Landscape" is to be summarized, encompassing the general idea of machine learning, the types of machine learning, applications, and main challenges.

Machine learning is the science that provides computers with the ability to learn from data given without being explicitly programmed to. It shines when facing complex problems in an everchanging environment that requires continuous fine-tuning, these attributes can occur separately as each problem has unique circumstances. In addition, machine learning has been used to detect insights that may have gone unnoticed otherwise (e.g. Checking undetected sentences that may make up a spam email).

When relating to some diverse applications, they include:

- Image Analysis
- Tumor Detection
- Revenue Forecasting
- Document Summarization
- Chat-bot Creation
- Flagging Offensive Comments
- Credit Card Fraud Detection

Machine learning systems are classified according to different criteria.

- How are they supervised?
- Can they learn incrementally?
- Do they compare new data to known ones or build predictive models to learn?

Supervision relates to how the machine learning system learns the training data. Systems that take the training set with known solutions (also known as Labels) are **Supervised**, this type of learning is commonly used for classification (Categorical prediction) and regression (Numeric Prediction) problems, in the contrary, **Unsupervised** learning aims to understand the inherent structure of the data, thus not having the access to solutions with the set. Clustering data (e.g. grouping them into categories) is a famous application of unsupervised learning.

Other types can be addressed:

- **Semi-supervised** learning is fed partially labeled training sets and can be done via a combination of supervised and unsupervised where data is clustered at first, then labeled to the nearest commonly labeled, ready to be used for supervised learning tasks.

- **Self-supervised** learning transforms an unlabeled dataset into a labeled set, an example would be having images of animals and masking them while maintaining another set representing the original set with the goal of re-obtaining the original set from the newly masked set. This type is usually not the end goal but a process that might be used for other tasks.

- **Reinforcement** learning involves an agent that interacts with the surrounding environment performing actions as it sees fit while being rewarded either positively or negatively to find the best action to do (known as a Policy).

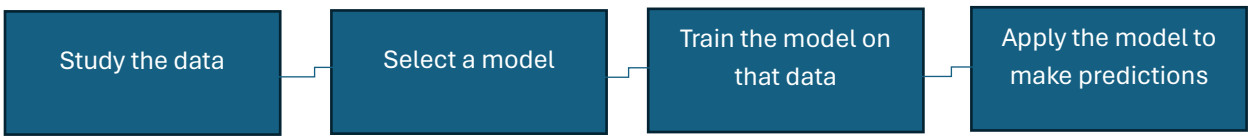
The ability to learn incrementally is another criterion a machine learning system falls into, a **batch-learning** system can not learn incrementally as it learns the training data once and is launched without the ability to learn later on. Batch learning is also called offline learning and suffers from “model rot” as the data remains still in comparison to the changing technological landscape.

On the other hand, **online learning** systems can be trained incrementally, they learn the data sequentially (called mini-batches) and in steps that are usually fast and cheap. Online learning is a good option for systems that need rapid changes and have limited computing resources. They provide “out-of-core” learning that splits a massive dataset into parts and runs them in steps repeatedly if it cannot be processed in a single iteration.

The last criterion is based on how the model generalizes the newer data points**. Instance-based** systems learn the training data by heart and compare new data points against them to test them in similarity. **Model-based** systems learn the data and build a model based on them to make predictions.

Note: The above criteria are not mutually exclusive, a machine learning model can be supervised, online learning, and model-based for instance.

To summarize the basic machine learning workflow for now:



Challenges that face machine learning are a result of two main culprits: bad data and/or bad models.

Data can be:

- Insufficient training data in terms of quantity.
- Non-representative of data that affects the accuracy.
- Poor Quality data with errors, outliers, and noise.
- Full of irrelevant features

A model, however, can encounter many problems that make it bad. When not chosen carefully it can **overfit** data, as in perform great on training data but cannot generalize actual new data quite well. **Overfitting** can be constrained using **regularization** which aims to simplify and reduce its risk, and that in turn is controlled by a **Hyperparameter** configured before the training phase. **Underfitting** happens when the model is too simple to learn the underlying structure of the data. Selecting a more powerful model, better features, or reducing the constraints, are all valid options to fix it.

Following the training phase, the evaluation and fine-tuning phase commences. When speaking of one model you would go through with **testing and validation.**

- Split the dataset into a training set and a testing set.
- Focus on the generalization error on the new cases to get a feel for how the model may perform on never seen before cases.
- One particular case is that if we observed a very low error training error earlier on and a high generalization error later on, we would suspect that the model is overfitting.

This evaluation is simple, think of the case that we are to choose between two models and we choose the one that generalizes better and we would want to choose a regularization hyperparameter but how?

For Hyperparameter Tuning and Model Selection, A holdout set called the validation set or the dev set is withheld from the full training set.

- Multiple models are trained on the reduced training set.
- The best model that performed on the validation set is selected.
- The model is hence trained on the full training set again.
- The model is evaluated against the test set for the generalization error.

This solution works well but is affected by the size of the validation set, thus introducing the cross-validation technique. It uses many validation sets for models to be evaluated against and by averaging out all evaluations you get an accurate measure of performance, but takes up more training time as it is multiplied by the number of validation sets.

In some cases, the data collected can be large yet non-representative of the data needed in actual production. This is **Data Mismatch.** We would want both the validation set and the testing set to be as representative as possible to get results close to what is expected in production. The problem arises after training, in the case of poor performance, where we cannot know if overfitting or data mismatch is the direct cause. The solution follows:

- Another dataset is withheld called the train-dev set.
- The model that performed poor after on the train-dev set means that it suffers from overfitting.
- The model with good performance would be evaluated then on the validation set where it could perform poorly (Data Mismatch) or greatly where it moves on to the testing set.

Not part of the book

Machine learning Idea : A Smart Compost Helper

- This mobile application would use a smartphone's camera and machine learning to detect and identify scraps and waste. Upon viewing a compost pile, the app would analyze the contents and offer advice on how to get rid of it.

- Machine Learning Aspect: Since the app is trained on large datasets of images containing different types of compostable materials. It has the ability to recognize what's in the compost via image recognition.

- Basic and Unique: Composting is a household practice commonly encountered, but can be confusing at times. This application uses machine learning in a simple and approachable way to improve the home composting experience.