Twitter Thread by Quant@Python

Quant@Python
@BeingHorizontal

Thread for beginners; How to learn machine learning.

Let’s start with something really simple and useful not just for Machine Learning.

Start with a Python library called pandas. It’s a huge library but it's mostly fun for traders because it's kind of rewarding to learn each function and how it gonna affect the way you handle the financial data.

Then learn some basic stuff about Python library called NumPy. Pandas is built on NumPy. Numpy is huge, you just need to understand some basic NumPy functions like NumPy shapes and NumPy arrays

If you understand pandas then you will understand NumPy arrays and shapes easily. Side note: Numpy shapes gonna play an important part in the deep learning model called LSTM - mainly used for time-series analysis(such as financial data) and Natural Language Processing (NLP)

Next. Install python library sklearn (pip install sklearn)

Next. Try to understand what the stationary and not stationary data is. Try to find the intuitive meaning of it and not a theoretical one.

Understand what data normalization is. Take your own sweet time. It’s not hard but it is really important to understand what it is. Most of the new folks tend to make mistakes here when it comes to real-life application of machine learning.

Again try to find the intuitive meaning of above and don’t get bogged down by formula and technical explanation. Especially academic debate on what should be called as stationary data is a mind fu*k, lol. You can completely skip that technical part and no one will sue you.

Now the most important part. Try to understand what overfitting is. This is a huge issue in machine learning and there are no correct answers. Just understand why you need to split the data between train and test

For the sake of understanding, you can say training data is like in-sample data and test data is out-of-sample data or hold out data. Test data is often referred to as validation data as well, don't get lost in jargon.

Then there is a concept called k-fold cross-validation, where we split the data into various folds(k-fold) often randomly and check whether predictions on training dataset match with various validation sets (cross-validation)

Avoid k-fold cross validation if it makes you confuse but do note that it is one of the better methods to avoid overfitting (except LSTM, the answer is out of scope).

Then, learn what look-ahead bias is. It is a cheeky little bastard. For instance, if you're using the 'mean' of the whole data set and then splitting the data between train and test then you unknowingly added look-ahead bias in the form of 'mean'. See? Commonsense.

Okay mathematical part is done (unless it's not, lol)

Machine learning is divided into 2 main parts supervised and unsupervised. Unsupervised is useful when you want to classify the data. Supervised learning is useful when you want to predict the likely outcome.

Focus on supervised learning. Supervised learning models are further divided into 2 parts. Easy and hard. Yep, I'm making that up.

Hard: all the deep learning models which are based on gradient descent and matrix multiplication. For that, you need to install Tensorflow 2 which now comes with Keras and detects GPU support. Another alternative is Pytorch.

Since this thread is for beginners, start with the easy supervised learning model called Decision Trees (it comes bundled with sklearn that we installed early in the thread).

Don't get fooled by its simplicity of Decision Trees. They are quite robust and highly recommended if your data is a) Relatively small (non-tick data), b) Well classified (tabular) c) and Full of outliers. The bonus, you don't have to worry about data normalization.

Decision trees are highly visual.
Naturally, decision trees make an intuitive sense for learners. While you learn decision trees I highly recommend the Titanic survival prediction problem. It is something you can relate to which is quite useful for learning something new

Once you learn decision trees, then use their better variant called random forest which is nothing but a bunch of decision trees ensemble together. Also, another variant is gradient boosting trees performance wise they are at par with Random Forest.

Now you have to learn how to frame the problem.

You can frame the problem in such a way that model will predict whether the next day will be bullish or bearish- called the binary classification model (recommended) Or multiple classification models- next day will be bullish, bearish, neutral, upper-circuit, lower-circuit etc.

If you don't like the classification model then you can use a regression model. Forget your traditional definition of regression. Here regression models mean it will try to predict future value. Ex. Next day will be +2% or -4%

In my experience regression models are difficult to achieve accuracy. Especially if you are dealing with an extremely noisy data such as financial time series data.

Though I highly recommend playing with regression models so you will get familiar with how to use the activation function for classification and regression models.

Activation functions are useful to generalise the problem. For example, your machine learning model will not just remember a specific mathematical problem but instead, it will convert it into a formula and will use it in various different situations.

When you get comfortable with all of the above then it's time to learn deep learning.

Google 'gradient descent with figures' It is the backbone of deep learning.

Of course, you can skip all that and come back later. It really helps though to understand what is going on behind the scene. For instance, you will often hear people complaining about their model getting stuck in local minima, chances are you will also face a similar problem.

Another backbone of deep learning is matrix multiplication. It is not that important to understand Matrix multiplication but it is up to you whether you want to go deep into Deep learning or not (Pun intended)

In deep learning, the community is 'mainly' divided between deep learning for image recognition & deep learning for natural language processing(NLP).

Of course, there are other variants like deep Reinforcement learning. Also, there are Genetic algorithms(GA).

Image recognition problems — also called computer vision (CV) is kind of a solved problem now. They have become extremely efficient. Convolution neural net(CNN) is a go-to model for CV.

For financial time series, our go-to model is an LSTM (at least for me, though I have seen guys using CNN for time series prediction). Whichever deep learning model you choose first play with a toy example and establish the baselines.

Unlike image recognition, time-series analysis isn't a solved problem by any stretch of imagination so it is really important to establish the baseline so you can compare your real data with the toy data. It is kind of sanity check whether you're going on the right track or not.

Then there comes the reinforcement learning, which I’m particularly not a fan of as they are extremely prone to overfitting. Though if you find it useful then go ahead and learn that.

End of the thread.

PS. Expecting comments from experts so do check replies.

Full disclosure I am not an expert.