MACHINE LEARNING

* -OCR one of the first
* -first main =spam filter
* -ML is the art and science of programming comps so they can learn from data
* -for spam filter=> training set= mails,training instance= single mail,task (T)= flag a mail if spam or not
* Experience (E)=training data, performance afctor(P) =accuracy
* -program gets shorter, maintainable, easy to understand using ml (spam on own code, lot of if ifelse blocks)
* -helps human learn (spams have keywords but what are they)
* -using ml to dig in data and find patterns=data mining
* -types:-
  + Trained with human supervision (supervides,unsupervised,reinforced,semisupervised)
  + Learn like \*snap fingers\* (batch online)
  + Compare data with old or make patterns(instance vs model)
* Supervised-> we give the answers kinda during training eg classification lr,logistic r,svm,knn
* Unsupervised->unlabeld data k means,dbscan etc anomaly detection
* It is often a good idea to try to reduce the dimension of your train‐ ing data using a dimensionality reduction algorithm before you feed it to another Machine Learning algorithm
* Semisupervised learning->partially labeled data eg dbn, rbm
* Reinforcement-> learning ystem (agent) observe the environment, select and perform actions, and get rewards in return It must then learn by itself what is the best strategy, called a policy, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation.
* Batch (offline) learning->learn before then put in production and just performs never learn thereafter. If you want a batch learning system to know about new data (such as a new type of spam), you need to train a new version of the system from scratch on the full dataset (not just the new data, but also the old data), then stop the old system and replace it with the new one.
  + train a new system only every 24 hours or even just weekly
  + Also, training on the full set of data requires a lot of computing resources
* Online learning-> you train the system incrementally by feeding it data instances sequentially, either individually or in small groups called mini-batches
  + continuous flow
  + adaptable
  + huge data on a scale which doesn’t fit in a single computer
  + If you set a high learning rate, then your system will rapidly adapt to new data, but it will also tend to quickly forget the old data vica-versa
  + Data should be as clear as it can get or else performance drops
* d instance-based learning-> the system learns the examples by heart, then generalizes to new cases by using a similarity measure to compare them to the learned examples (or a subset of them)
* Model-based learning->Another way to generalize from a set of examples is to build a model of these exam‐ ples and then use that model to make predictions.
* How can you know which values will make your model perform best? To answer this question, you need to specify a performance measure. You can either define a utility function (or fitness function) that measures how good your model is, or you can define a cost function that measures how bad it is. For Linear Regression problems, people typically use a cost function that measures the distance between the linear model’s predictions and the training examples; the objective is to minimize this distance.
* you feed it your training examples, and it finds the parameters that make the linear model fit best to your data. This is called training the model
* If all went well, your model will make good predictions. If not, you may need to use more attributes (employment rate, health, air pollution, etc.), get more or betterquality training data, or perhaps select a more powerful model (e.g., a Polynomial Regression model).
* BASIC ALGO:-
  + Study data-> select the model->train it->apply the model
* 2 main challenge:- bad algo or bad data
* BAD DATA:-
  + Imagine your kid needs to learn this is an apple, but he’s practical so you have to show him different exapmles like green apple, red apple etc that “this is an apple and those too are apples” that’s machine learning
  + if the sample is too small, you will have sampling noise (i.e., nonrepresentative data as a result of chance), but even very large samples can be nonrepresentative if the sampling method is flawed. This is called sampling bias.
    - 1936 american election, landon to win landslide but lost
    - Exit poll questions were sent to rich elites and 25% answered meaing 1 in 4 was interested in plotics so they votes for landon
    - 2024 indian election exit poll
    - I have to create a model for edm music, I search yt it shows fred again, Skrillex etc but may not be the same for a person who don’t speak English or doesn’t recide in such a country eg ja[anese might be shown Yosuke yoikimatsu.
* Poor quality data:-
  + Noise errors outliers (inorganic chem)
  + Main thing
* Irrelevent features
  + A critical part of the success of a Machine Learning project is coming up with a good set of features to train on. This process, called feature engineering
    - Feature selection (selecting the most useful features to train on among existing features)
    - Feature extraction (combining existing features to produce a more useful one—as we saw earlier, dimensionality reduction algorithms can help)
    - Creating new features by gathering new data
* OVERFITTING:-
  + Overgeneralizing (someone ripped me off so all so all of th4 taxi drivers are scammers)
  + if the training set is noisy, or if it is too small (which introduces sampling noise), then the model is likely to detect patterns in the noise itself.
  + Constraining a model to make it simpler and reduce the risk of overfitting is called regularization. Eg there are 2 variables, make one veery small movable.
  + Solutions:-
    - Simplify the model by selecting one with fewer parameters
    - More data
    - Reduce the noise
* UNDERFITING
  + it occurs when your model is too simple to learn the underlying structure of the data. For example, a lin‐ ear model of life satisfaction is prone to underfit; reality is just more complex than the model, so its predictions are bound to be inaccurate, even on the training examples.
  + Here are the main options for fixing this problem: • Select a more powerful model, with more parameters. • Feed better features to the learning algorithm (feature engineering). • Reduce the constraints on the model (e.g., reduce the regularization hyperpara‐ meter).