+sklearn: small model (techniques)

-linear

-trees

Basic,simple, weak netwirk well

+ deeplearning NN : Ml resarch

Best for images, sounds (maybe for text) videos

Tuning parameters and architecture

+gradient boosting : lightGBM and XGB : in finance and other realise fields : Data are tabular (nombres, categorical….) and deep learning does not work wellwith this type of data

+key : creating new and useful features : tree-based gradient boosting is the best for this type of data but tree based need good features : Feature engineering is the most important thing

-Feature engineering : combining Several features : -> new one

-modfying features

-Titanic : #sibilings

#parents children

Family Size-🡪 Alone/ not alone

-Name : « Mr ….. »

« captain….. »

Found the title of that person

-Categorical-> Numeric data

-one-hot : (00….1…)=-> not elegant-🡪 too many features if you have k categories-🡪 k new features

Ex : name of the city (not categorical)

-Frequency : # times that category appear in data. Ex : Paris,Paris,lyon,Paris,Lyon

Paris = 3

Lyon = 2

Book : feature engineering by alice Zheng,o’relly 2018

Why feature engineering :

Tree-base learning

Work well with categorical data.

Don’t need to scale features

Bagging : independent

Build many trees and compute average / majority voting

Bagging plus ou moins égal au kandan forest

But random forest : trees on subset of freatures

Bagging : trees on all features

Boosting plus ou moins égal baging :

Build tree 1-> build tree 2 based on the error of tree 1

Build tree 3 based on the error of tree 2

* Weighted average.

Bossting is better than bagging in avevage but boosting can be overfitting.

Boosting ang bagging can be used for any technique (not necessary a tree)

.linear

.logistic reg

State of the art :

Tree-based gradient boosting

Libraries : light GBM (2016)

XGB(2015)

Light GBM is faster and generally better

LightGBM : lightgbbm.readthedoc.io

XGB :Xgboos.readthedoc.io

Usage : simple : no architecture

Just tuning parameters

3 choice : cat boost

Pipeline :

Final step :

Blending

Stacking

Your train on many different models

* Get many different predictions

lazy people <- blending ; final prediction (Yfinal)= weighted average (y1,…..yt)

Stacking : use y1,yt as t features to train a final, simple model and return Y final. Y final = model(Y1, Yt….)

Whole ML pipeline.