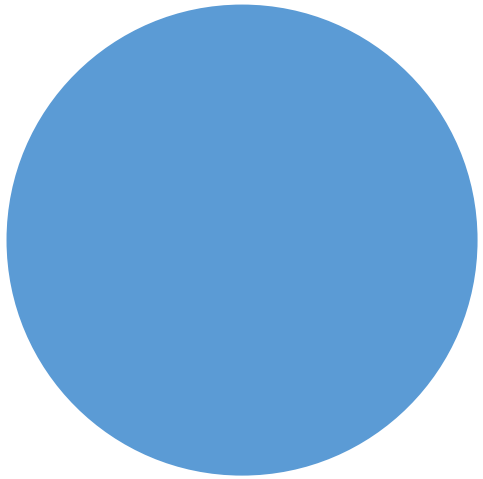
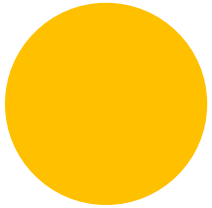
Data science toolbox

Author: Maxim Panteleev

Outline

- Common ML workflow
- Not beyond sklearn
- All we need is pandas. Or not ☺
- Handling categorical variables
- Gridsearch and hyperopt
- Power of crowd
- The sense of trees xgboost, lightgbm

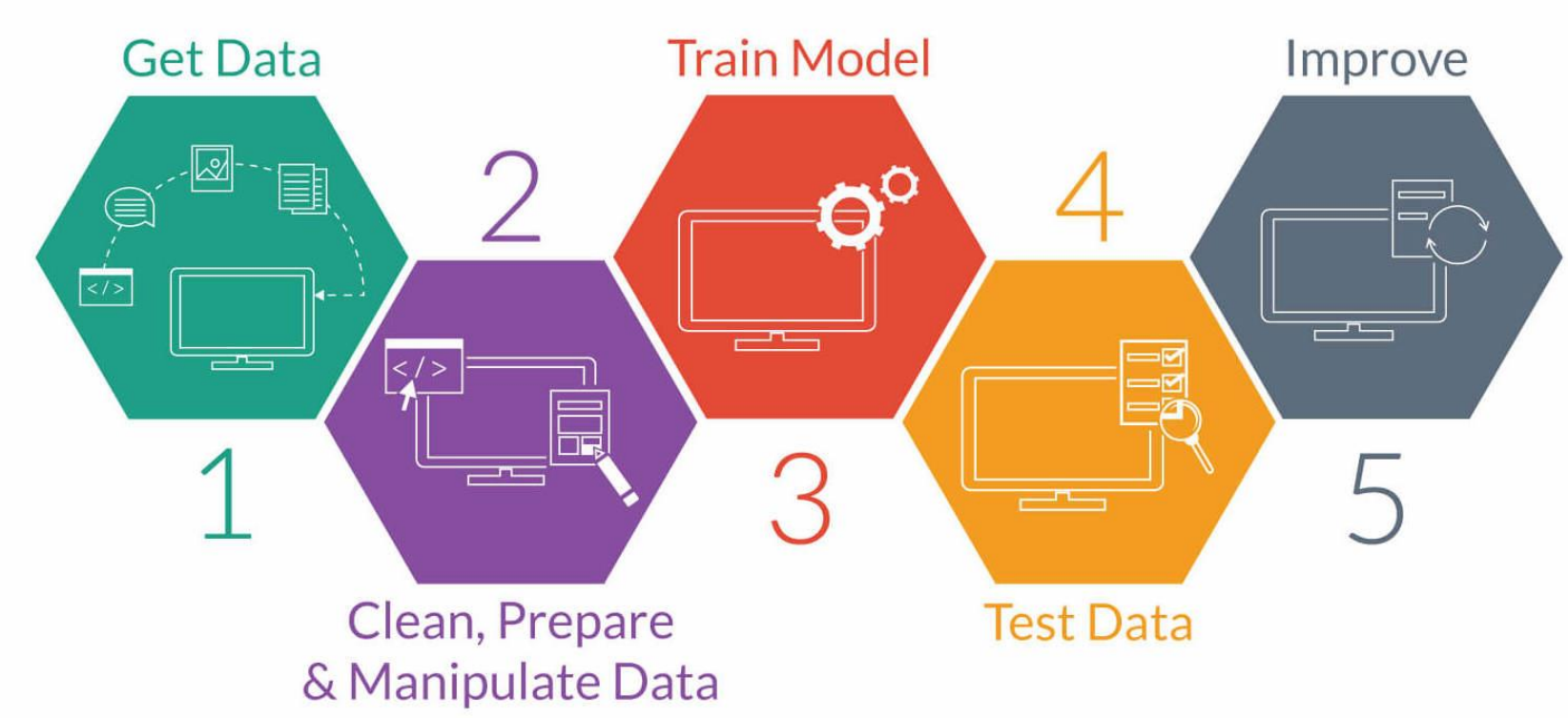



Common ML workflows

How we usually build models

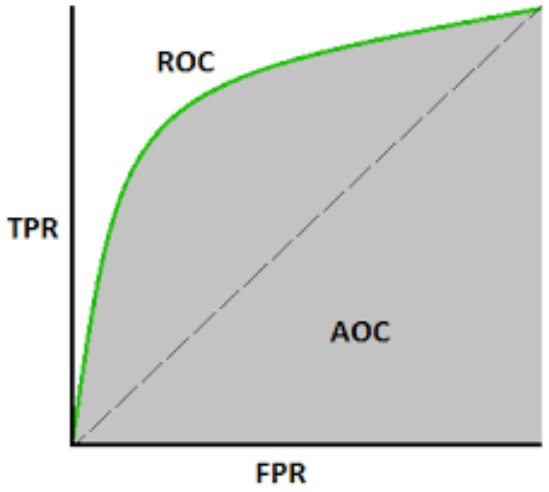
Common ML workflow

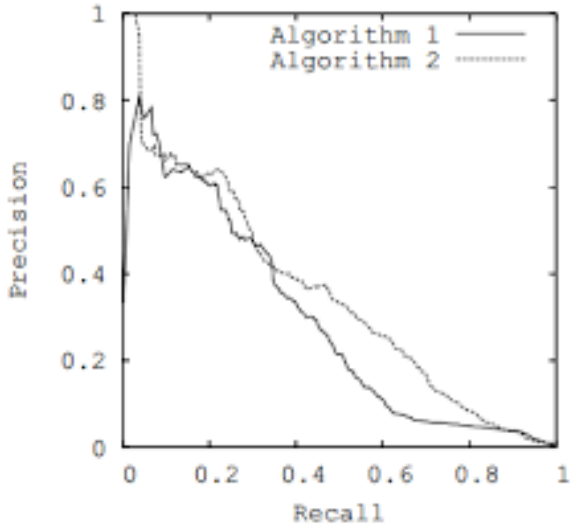
- Define metric and …



Common ML workflow: metrics

- From the first principles – MLE/MAE – (r)mse, mae, mape, log loss
- Popular: precision, recall, roc/pr –auc, F1 score
- Or it depends on business needs





$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

$$specificity = \frac{TN}{TN + FP}$$

Common ML workflow: data engineering

- numpy
- Pandas
- sklearn
- Dask
- Nvidia rapids
- Tensorflow, pytorch, Theano

- Pssss! Bigdata – spark, Hadoop, flink, etc

Common ML workflow: models

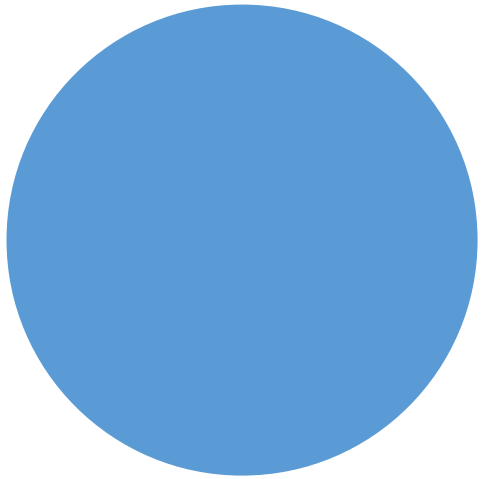
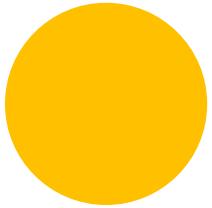
- Sklearn
- Beyond sklearn (xgboost, lightgbm, catboost, etc)
- Even different library (e.g. H20)

- General advice: pick what you are familiar with most

Common ML workflow: optimization

- Metrics optimization:
- 1) Setup right cross-validation procedure
- 2) Stratified Kfold, timeseries folds, choose number of folds

- Performance optimization:
- 1) memory/train-predict time
- 3) adjust to the different computing environment: cpu, gpu, specialized hardware

Sklearn overview

Conquer the world with the only one weapon

Sklearn: common ML tools

- Local on-CPU training
- Feature engineering
- Pipelines
- Common API
- Models:
  1) supervised – classification and regression
  2) unsupervised – clustering
- Dimensionality reduction
- Model selection and hyperparameter optimization
- Routines and primitives for text processing (very simple)

Sklearn: common API

- Supervised API:
  fit(X, y) – train, predict(X, y)/predict_proba(X, y)

- Transformer API
  fit(X, y), transform(X, y), fit_transform(X, y)

- Cluster API
  fit(X), fit_predict(X), fit_transform(X, y)

Sklearn: pipelines

- BaseEstimator:
  set_params(**params), get_params()

- Pipeline
  1) meta estimator with fit/predict/transform/etc
  2) internally it is a set of steps, the goal is to apply multiple functions (steps) to input

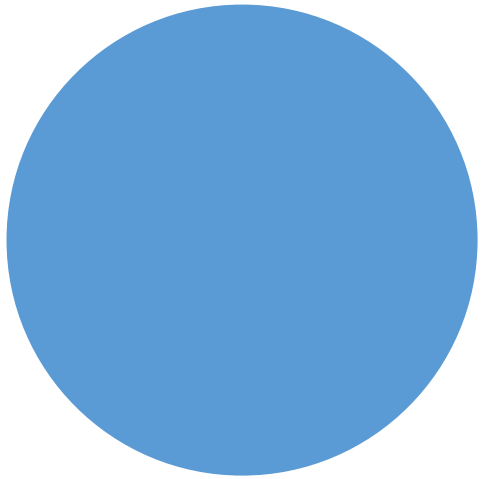
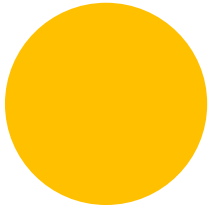
- FeatureUnion
  combine multiple data sources altogether

Sklearn: common API

- Supervised API:
  fit(X, y) – train, predict(X, y)/predict_proba(X, y)

- Transformer API
  fit(X, y), transform(X, y), fit_transform(X, y)

- Cluster API
  fit(X), fit_predict(X), fit_transform(X, y)

Data engineering

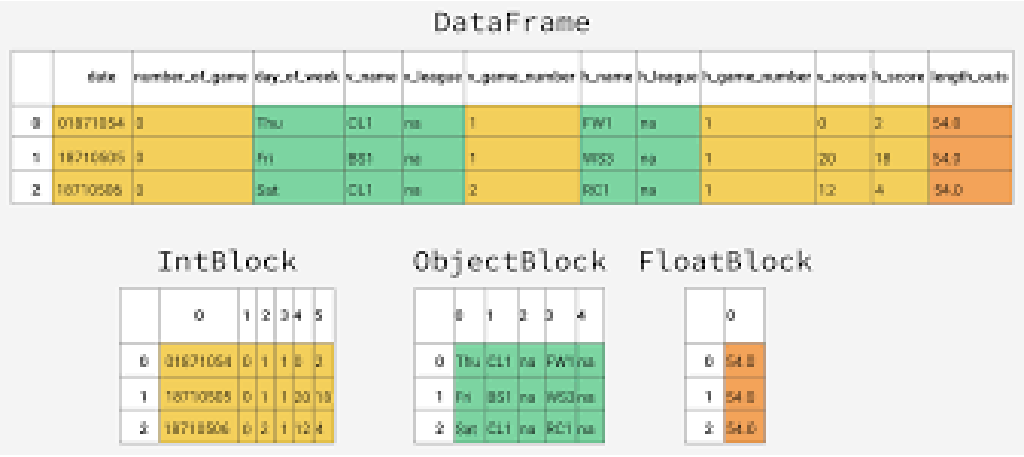
Data loading, feature extraction, transformation

Data engineering: numpy and pandas

- Numpy: ndarray, integer based indexes, mask
- Pandas: extending numpy with non-integer indexes, table like structure

Data engineering: optimize memory layout

- Pandas memory layout
- Axes: index, columns
- Block manager keeps track of block locations
- Block is a collection of columns with the same dtype



Data engineering: optimize data load

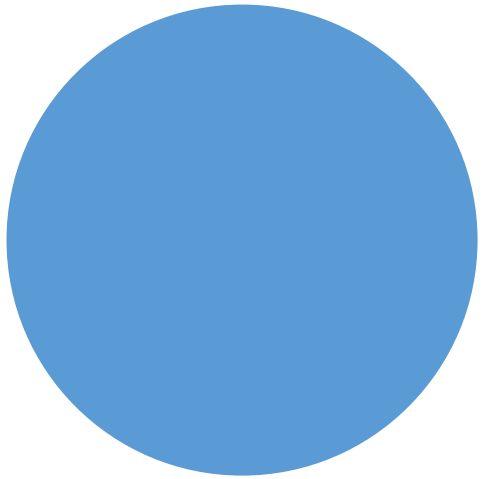
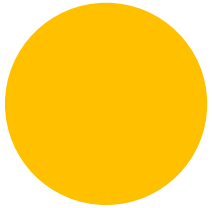
- Pandas data load
- pandas.read_’whatever’
- Whatever: csv, url, excel, parquet, etc

Data engineering: optimize memory layout

- Pandas memory layout
- Axes: index, columns
- Block manager keeps track of block location
- Block is a collection of columns with the same dtype

Data engineering: optimize performance

- Numba (JIT), swifter
- Install openblas, lapack, atlas libraries
- Joblib – going multiprocess

Going categorical

Transform red, green apples to vectors

Categorical variables

- Category types: text (words), colors, texts taxonomy, animals and other enums, people classification by weight, height, age
- They could be comparable: baby < adult (age) - ordinal

- The goal to transform them to numbers

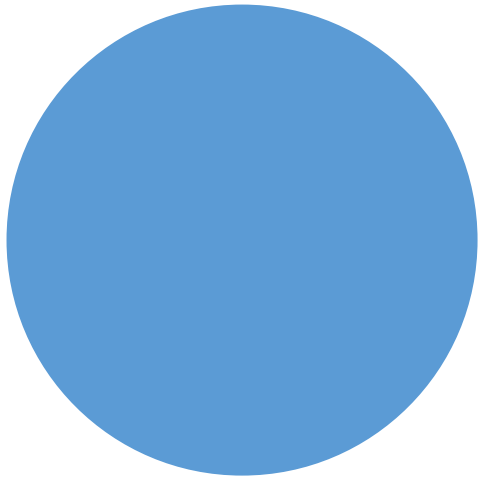
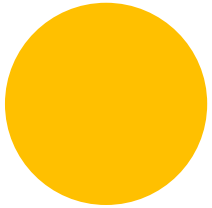
Categorical variables: encoding

- One-hot (dummy) encoding – sparse or dense representation
- Hash encoding
- Binary encoding
- Target encoding
- CatBoost encoding
- Cat2vec encoding

- How to choose?

Categorical variables: dependencies

- The problem: a lot of categorical features and we need to choose the best N or remove duplicates
- Chi2
- Cramer’s V – adjusted chi2
- Uncertainty coefficient (Theil’s U) – how much information we get about Y given X

Optimizing hyperparameters

When we want more

Optimizing hyperparameters

- When we want to get maximum performance according to chosen metric
- The old and proven way – gridsearch
- GridSearchCV
- RandomizedGridSearch
- hyperopt

Optimizing hyperparameters: gridsearch

- Parameter grid
- model
- Evaluation metric
- Cv-params
- Fit

- Any problems?

Optimizing hyperparameters: randomized gridsearch

- Parameter grid
- model
- Evaluation metric
- Cv-params
- Fit

- Any problems?

Optimizing hyperparameters: hyperopt

- Tree Parzen estimators - sequential model-based optimization
- Expected Improvements (EI) over surrogate p(y|x)

$$\mathrm{EI}_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) p(y|x) dy$$

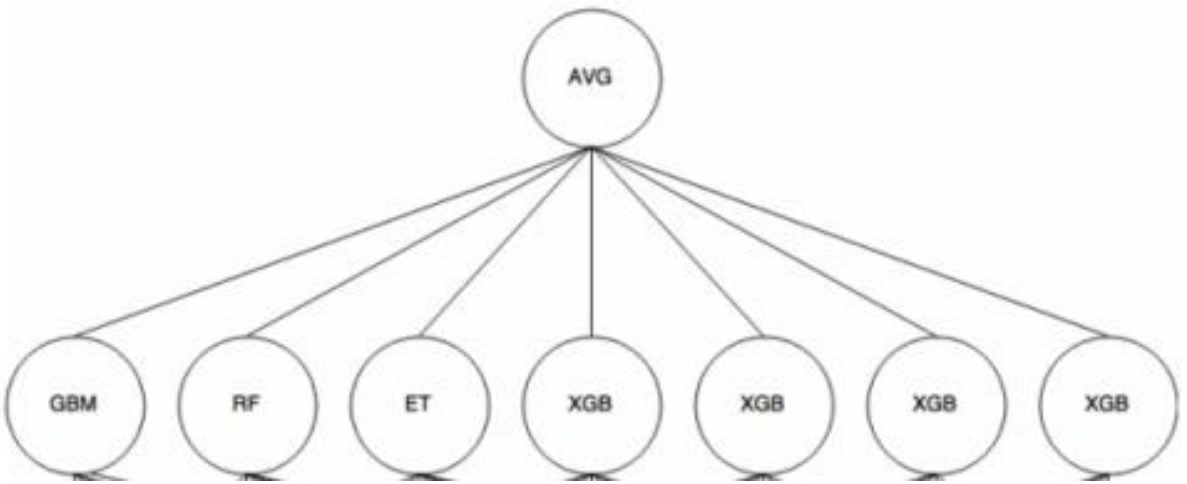
$$p(x|y) = \begin{cases} \ell(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases}$$

Power of the crowd

When we want even more

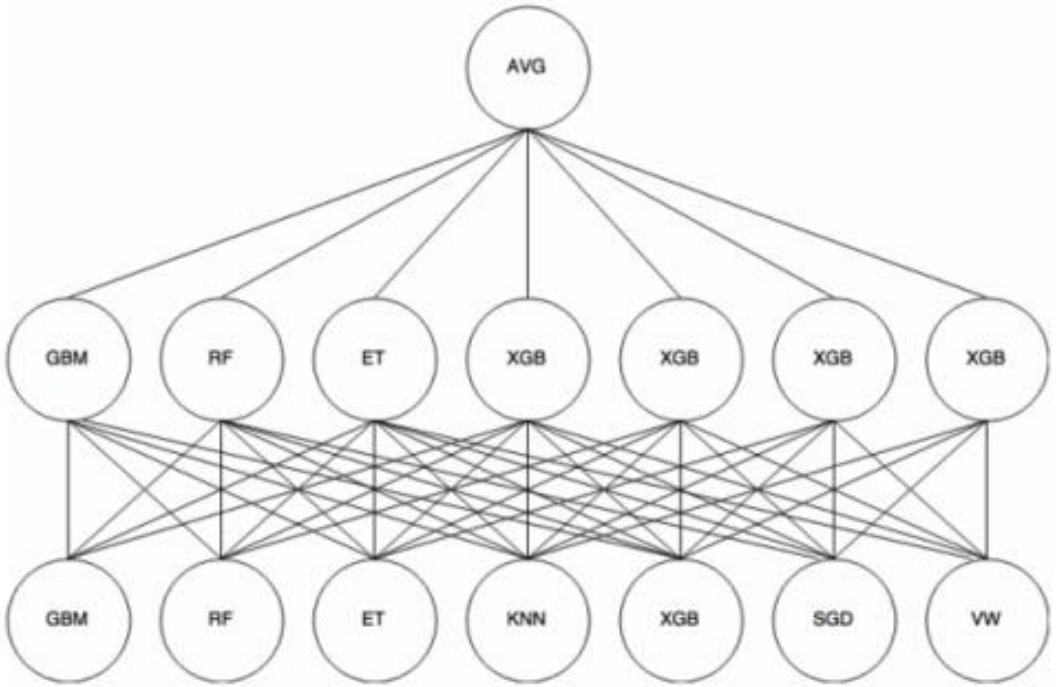
Power of crowd: Ensembles

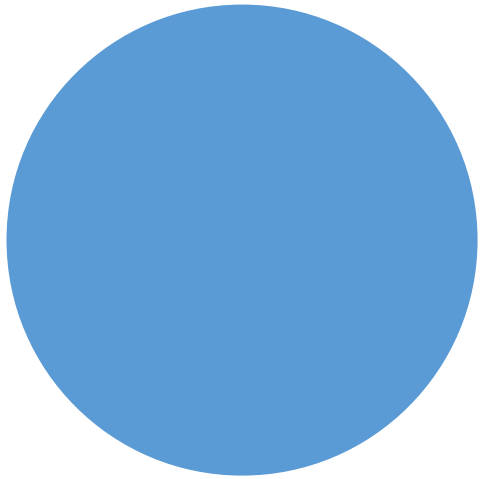
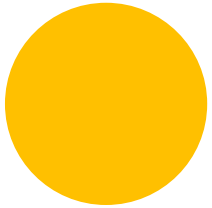
- Come up with multiple different models*
- Combine results of each model (soft with weight, hard)
- Example: random forest



Power of crowd: stacking and blending

- Come up with multiple different models*
- Combine results of each model (soft with weight, hard)
- Example: random forest



Boosting Trees

When we want even more

Boosting trees

- Xgboost was the most popular.
  Without embedded categorical encoding
- Lightgbm (Microsoft) is faster but tends to overfit – tune max_depth and num_leaves
  With embedded categorical encoding
- CatBoost (Yandex)
  With embedded categorical encoding

Thank you