DCAFPILOT FOR CMS

PILOT PROJECT FOR CMS COMPUTING DATA-MINING

by Valentin Kuznetsov / Cornell University

TALK OUTLINES

- Project scope
- Project outlines
- From data to prediction
  - gather information from data-services
  - prepare information suitable for data analysis
  - train learner algorithm
  - make prediction
- Future direction

PROJECT SCOPE

- DMWM-Analytics (cms-dmwm-analytics@cern.ch) group would like to improve our understanding of CMS computing data, full list of projects: https://twiki.cern.ch/twiki/bin/viewauth/CMS/CMSComputingAnalytics
- Ultimately we'd like to learn from CMS data and make prediction to improve our resource utilization.
- Initial goal is to predict popularity of new datasets.
- Start with understanding metrics, analysis workflow, tools:
  - DCAFPilot (Data and Computing Analysis Framework) is a pilot project to understand machinery involved with this problem.

WHY WE NEED THIS

- CMS has Dynamic Data Placement group which uses historical information to place popular dataset to sites
- We would like to predict which datasets will become popular once they appear on a market
- We can extend the scope of previous task to predict decline in popularity of certain datasets, reduce redundant activity, improve resource allocation, etc.

PROJECT OUTLINES

The DCAFPilot consists of several components:

- Dataframe generator toolkit: collect/transform data from CMS data-services (DBS/PhEDEx/SiteDB /PopularityDB/Dashboard) and extract necessary bits for datasets in questions

- Machine Learning (ML) algorithms (python/R code) for data analysis

- Data manipulation scripts: merge, transform, check predictions, etc.

Get the code:

```
git clone git@github.com:dmwm/DMWMAnalytics.git
```

Dependencies:

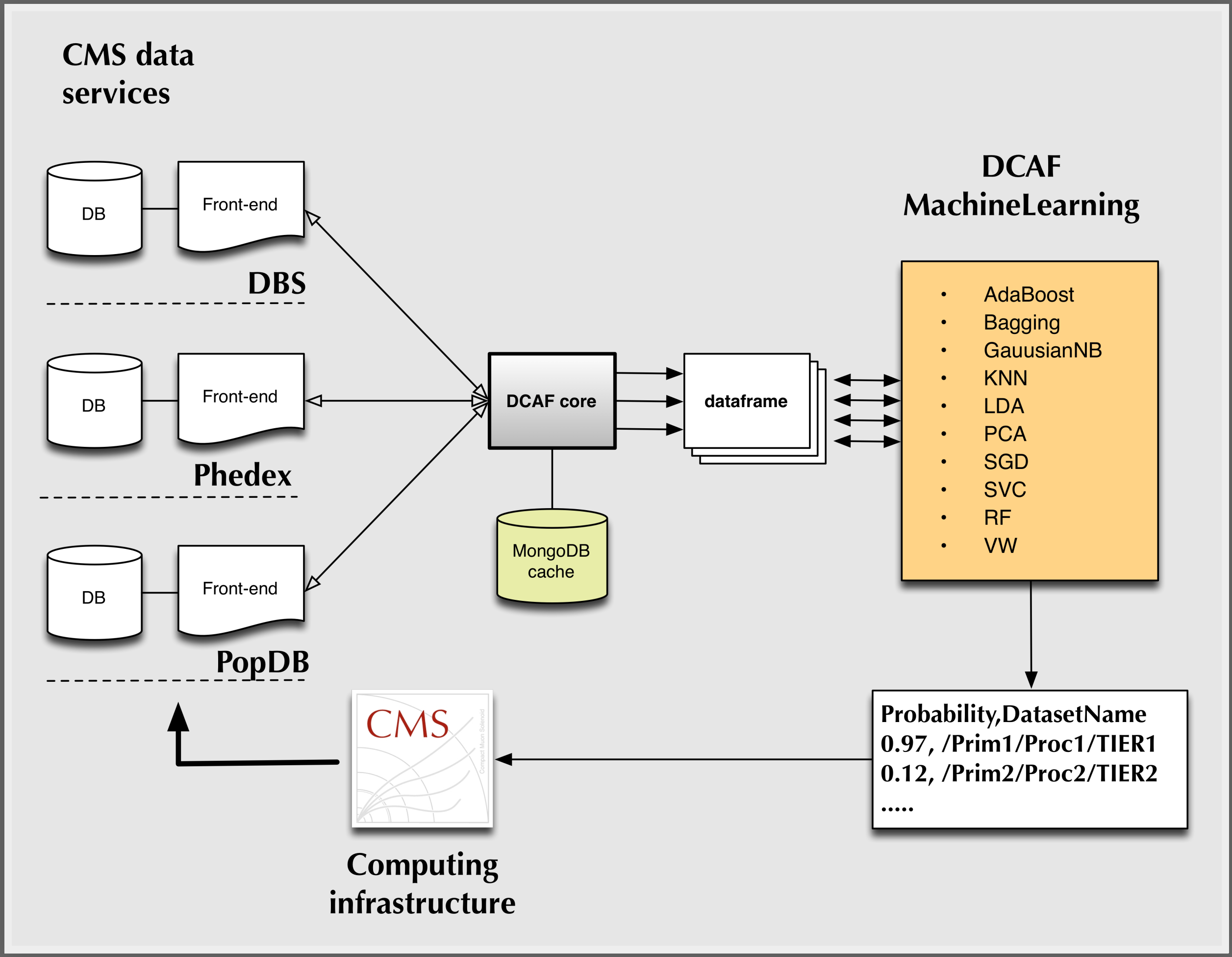
- DCAFPilot project, available at github
- MongoDB for internal cache
- Python tools: pandas, NumPy, SciPy, sklearn ML toolkit
- Optional: R language for data exploration and ML algorithms, SKLL toolkit to run/experiment with various ML algorithms, any other external ML toolkits, e.g. Vowpal Wabbit online-learning algorithm

DATA COLLECTION FLOW

- Collect data via the following set of rules
  - Collect all datasets (4 DBS instances) into internal cache
  - Collection popular datasets from PopularityDB on weekly basis
  - Get summary for datasets from DBS/PhEDEx/SiteDB/Dashboard data-services
  - Complement dataframe with random set of DBS datasets which were not visible in popularity for given time interval

- CMS data-service APIs used by DCAFPilot package
  - DBS: datasets, releases, filesummaries, releaseversions, datasetparents APIs
  - PhEDEx: blockReplicas API
  - SiteDB: site-names, people APIs
  - PopularityDB: DSStatInTimeWindow API
  - Dashboard: jobefficiencyapi API

DATA COLLECTION FLOW DIAGRAM



DATAFRAME PREPARATION, CONT'D

```
Queried 5 CMS data-services: DBS, PhEDEx, SiteDB, PopularityDB, Dashboard
   - used 10 APIs to get data content
   - feed internal cache with ~220K datasets from 4 DBS instances,
     ~900 release names, 500+ site names, ~5k people DNs.
   - placed ~800K queries

The final dataframe is constructed out of 78 variables and has 52 files and ~600K rows
   - each file is worth of 1 week of CMS data, ~600KB zipped/file
   - each file has about ~1K of popular datasets plus 10K random "un-popular" datasets

Elapsed time: ~4h to 1h per job, times fade out due to cache usage (MongoDB)
All jobs run on two CERN VM w/ N jobs/core splitting
```

We anonymized all data and performed factorization via internal cache

```
id,cpu,creator,dataset,dbs,dtype,era,naccess,nblk,nevt,nfiles,nlumis,nrel,nsites,nusers,parent,primds,proc_evts,procds,rel1_0,rel
999669242,207737071.0,2186,20186,3,0,759090,14251.0,6,21675970,2158,72274,1,10,11.0,5862538,335429,30667701,373256,0,0,0,0,1,1,0,
332990665,114683734.0,2186,176521,3,1,759090,21311.0,88,334493030,32621,86197,1,4,8.0,6086362,968016,123342232,1037052,0,0,0,0,1,
....
```

2014 dataset is available at https://git.cern.ch/web/CMS-DMWM-Analytics-data.git

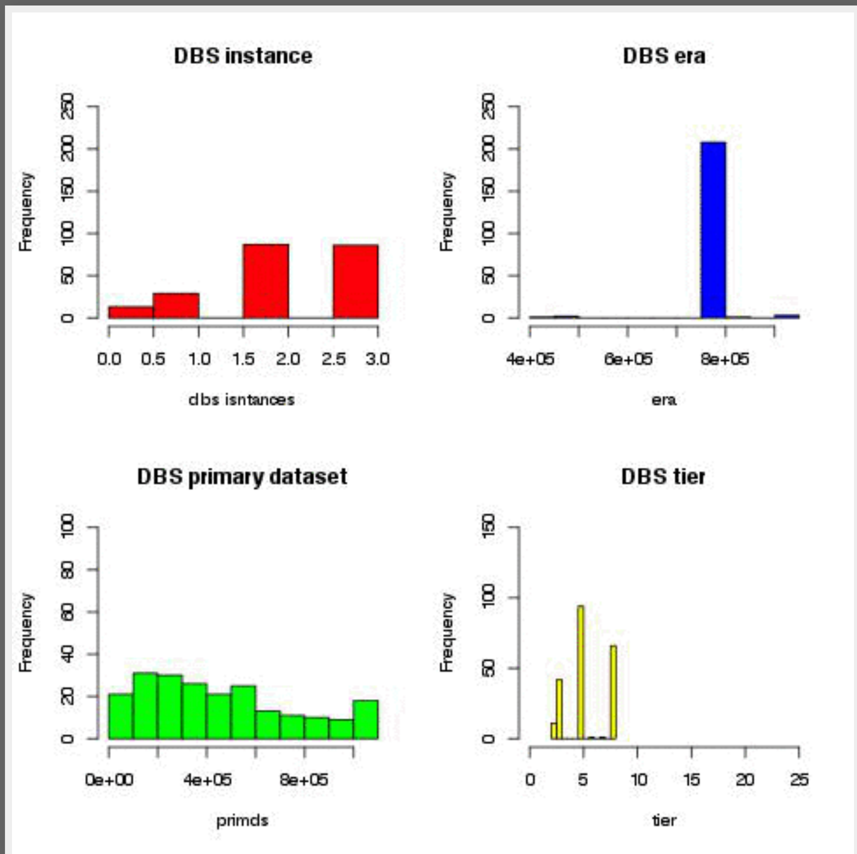
DATAFRAME DESCRIPTION

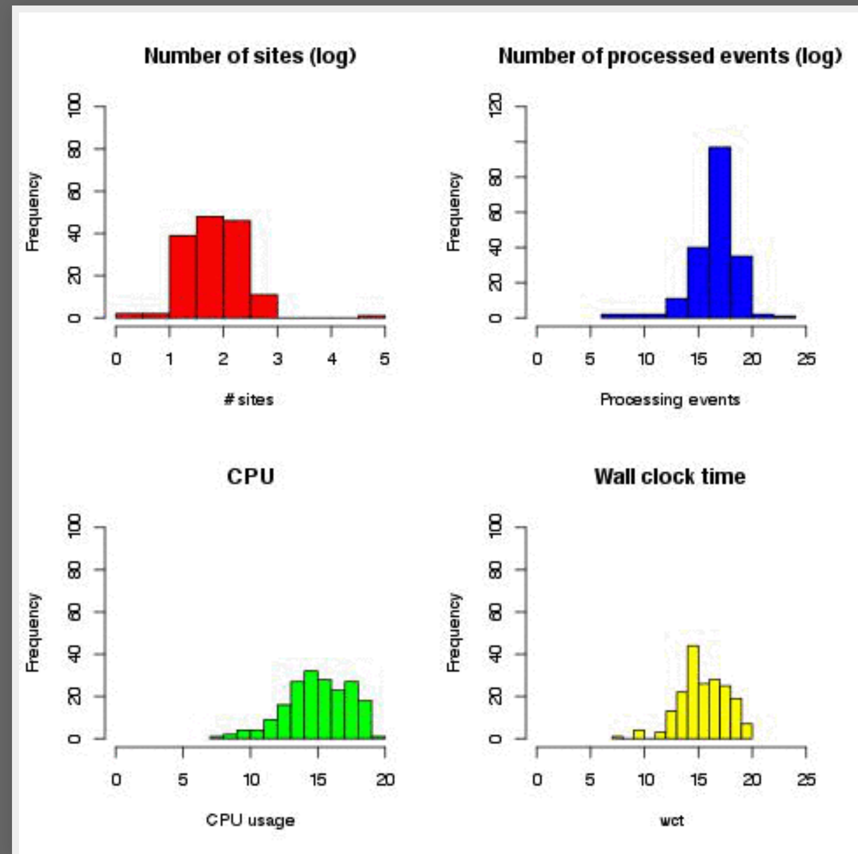
```
id: unique id constructed as long('%s%s%s'%(tstamp,dbsinst,dataset_id)) % 2**30
cpu: CPU time reported by Dashboard data-service for given dataset
creator: anonymized DN of the user who created given dataset, reported by DBS
dataset: DBS dataset id (comes from DBS APIs/database back-end)
dbs: DBS instance id
dtype: anonymized DBS data type (e.g. data, mc)
era: anonymized DBS acquisition era name associated with given dataset
nblk: number of blocks in given dataset, reported by DBS
nevt: number of events in given dataset, reported by DBS
nfiles: number of files in given dataset, reported by DBS
nlumis: number of lumi sections in given dataset, reported by DBS
nrel: number of releases associated with given dataset, reported by DBS
nsites: number of sites associated with given dataset, reported by PhEDEx
parent: parent id of given dataset, reported by DBS
primds: anonymized primary dataset name, reported by DBS
proc_evts: number of processed events, reported by Dashboard
procds: anonymized processed dataset name, reported by DBS
rel1_N: DBS release counter defined as N-number of series releases associated with given dataset
rel2_N: DBS release counter defined as N-number of major releases associated with given dataset
rel3_N: DBS release counter defined as N-number of minor releases associated with given dataset
s_X: PhEDEx site counter, i.e. number of Tier sites holding this dataset replica
size: size of the dataset, reported by DBS and normalized to GB metric
tier: anonymized tier name, reported by DBS
wct: wall clock counter for given dataset, reported by Dashboard

Target variables:
naccess: number of accesses to a dataset, reported by PopularityDB
nusers: number of users*days to a dataset, reported by PopularityDB
totcpu: number of cpu hours to accessed dataset, reported by PopularityDB
rnaccess: naccess(dataset)/SUM_i naccess(i), reported by PopularityDB
rnusers: nusers(dataset)/SUM_i nusers(i), reported by PopularityDB
rtotcpu: totcpu(dataset)/SUM_i totcpu(i), reported by PopularityDB
```

Some variables are useful for online learning while other can be used in offline context.

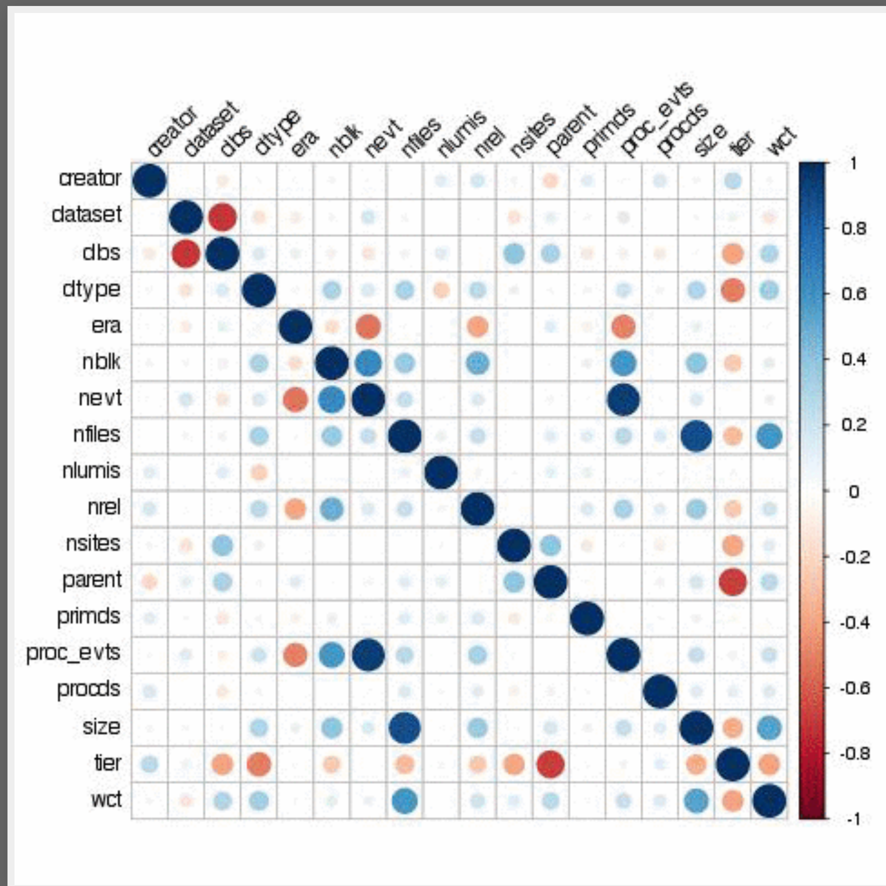
LIVE DATA PLOTS





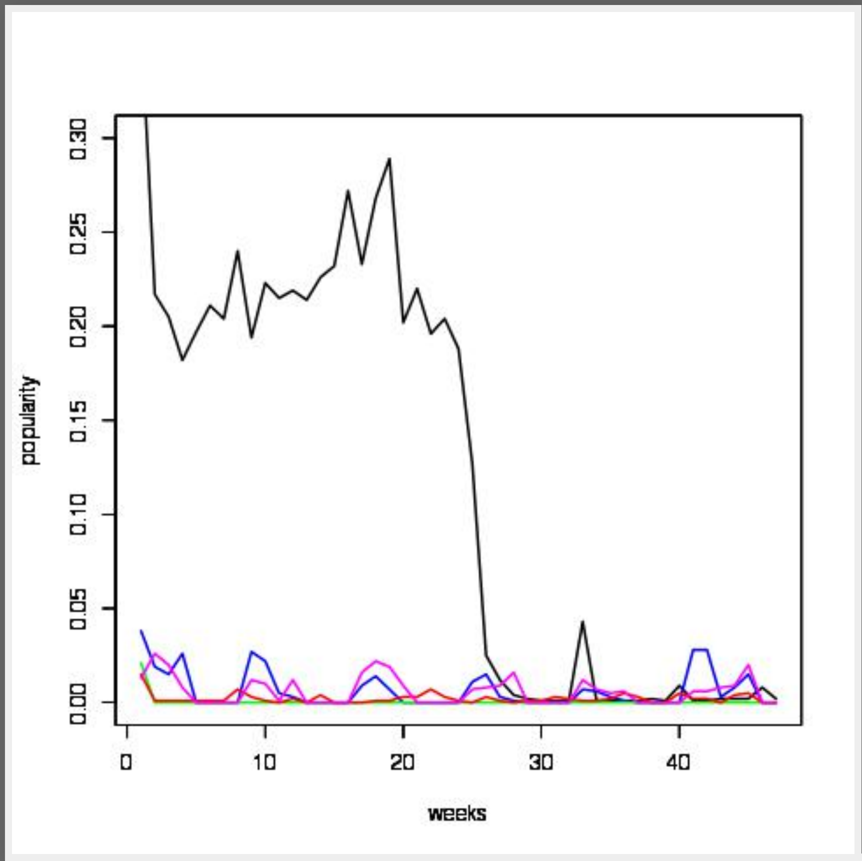
Data transition through 2014 on weekly basis

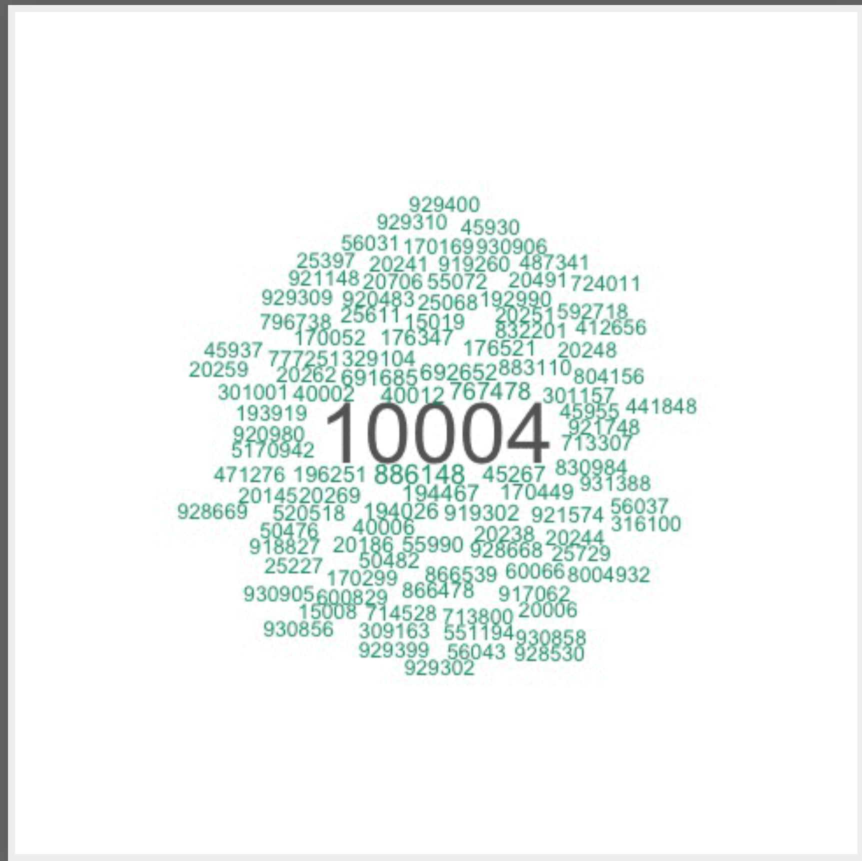
CORRELATIONS



Subset of variables, showing all of them in single plot can hard to swallow.

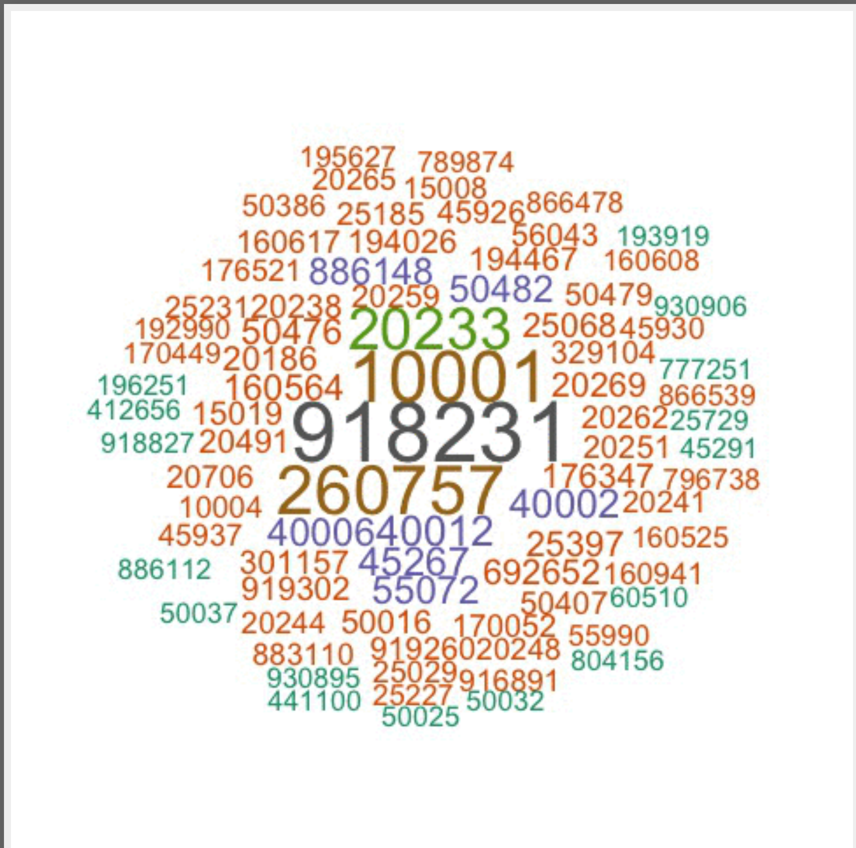
DATASET POPULARITY





Left plot shows few random datasets, while right one summarizes 100 most accessed datasets through 2014.
Observation: dataset access is like stock market, but N(datasets) >> N(stocks @ NASDAQ)

DIFFERENT DATASET POPULARITY METRICS



Left: popular datasets by nusers, Right: popular datasets by totcpu metric.
Therefore, target defition should be clearly defined. For the rest of slides I'll stick with naccess metric.

HOW TO ATTACK THIS PROBLEM

This seems to be time series problem, i.e. dataset popularity change over time

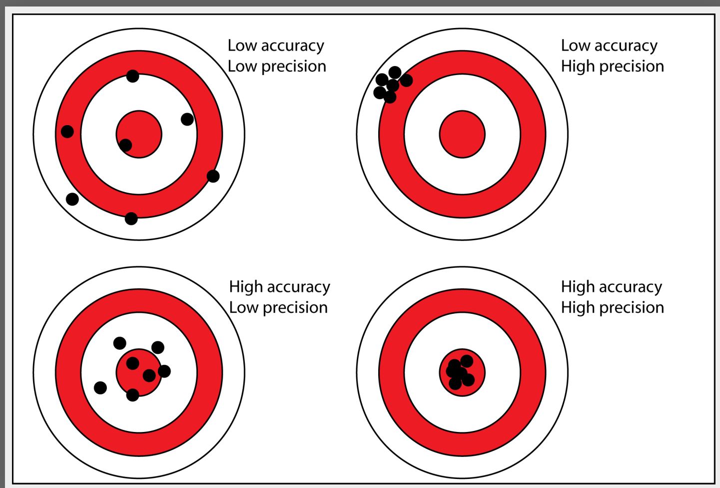
We can use rolling approach, like weather forecast, but for new datasets we do not have historical information.

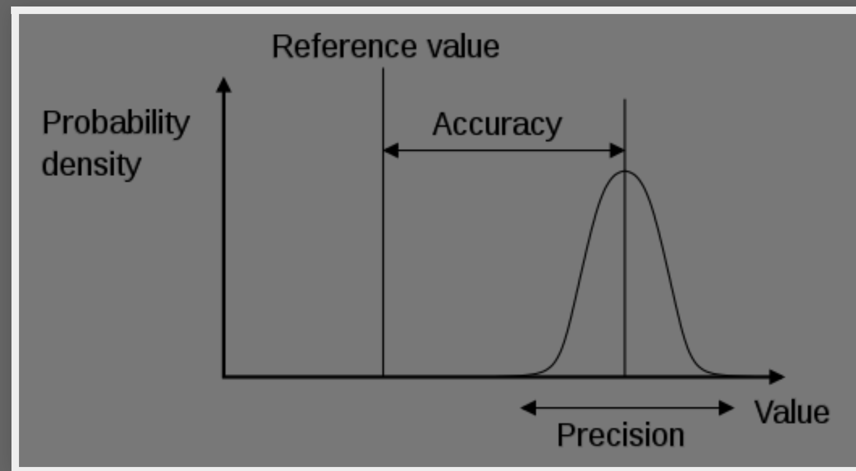
We can use either regression or classification approach. The former will allow to predict real values of metrics, e.g. naccess, while later can only classify between categories, i.e. popular or not. The classification will give us maximum gain with minimal inefficiency.

We can use any tools, e.g. python, R, online-learners, custom algorithm.

We'll show how to address this problem via DCAFPilot tools and discuss all steps from getting information to making prediction. We'll conclude about usefulness of the model by using a few statistical variables: accuracy, precision, recall and F1 scorers.

ACCURACY, PRECISION, RECALL AND F1





TP: true positive, TN: true negative, FP: false positive (false alarm), FN: false negative (miss)

$Accuracy=\frac{TP+TN}{TP+TN+FP+FN}$, $Precision=\frac{TP}{TP+FP}$,

$Recall=\frac{TP}{TP+FN}$, a.k.a sensitivity, fraction of relevant instances that are retrieved

$F1=\frac{2*Precision*Recall}{Precision+Recall}=\frac{2TP}{2TP+FP+FN}$, a.k.a weighted average of the precision and recall

FROM DATA TO PREDICTION

1. Generate dataframe or get it from existing repository
2. Transform data into suitable format for ML
3. Build ML model
   - use classification or regression techniques
   - train and validate your model
     - split data into train and validation sets
       we have ~600K rows in 2014 dataset
       train set (Jan-Nov), test set (Dec)
     - estimate your predictive power on validation set
4. Generate new data and transform it similar to step #2.
5. Apply your best model to new data to make prediction
6. Verify prediction with popularity DB once data metrics become available

FROM DATA TO PREDICTION, STEP 1-3

DCAFPilot tools: `merge_csv, model, check_prediction, pred2dataset`

```
# get the data, we keep it secure in separate CERN based repository
prompt_1$ git clone https://:@git.cern.ch/kerberos/CMS-DMWM-Analytics-data

# merge dataframes, then split 2014.csv.gz into train/valid datasets
prompt_2$ merge_csv --fin=CMS-DMWM-Analytics-data/Popularity/DCAFPilot/data/0.0.3 --fout=2014.csv.gz --verbose

# transform data into classification problem and remove useless variables
prompt_3$ transform_csv --fin=2014.csv.gz --fout=train_clf.csv.gz --target=naccess --target-thr=100 \
          --drops=nusers,totcpu,rnaccess,rnusers,rtotcpu,nsites,s_0,s_1,s_2,s_3,s_4,wct

# train the model
prompt_4$ model --learner=RandomForestClassifier --idcol=id --target=target --train-file=train_clf.csv.gz \
          --scaler=StandardScaler --newdata=valid_clf.csv.gz --predict=pred.txt

# check prediction
prompt_5$ check_prediction --fin=valid_clf.csv.gz --fpred=pred.txt --scorer=accuracy,precision,recall,f1
Score metric (accuracy_score): 0.982348203499
Score metric (precision_score): 0.79773214833
Score metric (recall_score): 0.952781844802
Score metric (f1_score): 0.868390325271

# convert prediction into human/CMS data format
prompt_6$ pred2dataset --fin=pred.txt --fout=pred.txt.out

# inspect prediction
prompt_7$ head -2 pred.txt.out
1.000,/GenericTTbar/HC-CMSSW_7_0_4_START70_V7-v1/GEN-SIM-RECO
1.000,/SingleMu/Run2012D-22Jan2013-v1/AOD
```

MAKING PREDICTIONS, STEPS 4-6

DCAFPilot tools: dataframe, transform_csv, model, pred2dataset, popular_datasets, verify_predictions

```
# seed dataset cache (w/ MongoDB back-end)
prompt_1$ dataframe --seed-cache --verbose=1

# get new data from DBS (you may need to run it in background)
prompt_2$ dataframe --start=20150101 --stop=20150108 --newdata --verbose=1 --fout=new-20150101-20150108.csv

# transform new data into classification problem similar to our train data
prompt_3$ transform_csv --fin=new-20150101-20150108.csv.gz --fout=newdata-20150101-20150108.csv.gz --target=naccess \
          --target-thr=100 --drops=nusers,totcpu,rnaccess,rnusers,rtotcpu,nsites,s_0,s_1,s_2,s_3,s_4,wct

# run the model with new data
prompt_4$ model --learner=RandomForestClassifier --idcol=id --target=target --train-file=train_clf.csv.gz \
        --scaler=StandardScaler --newdata=newdata-20150101-20150108.csv.gz --predict=pred.txt

# produce human readable format and inspect its output
prompt_5$ pred2dataset --fin=pred.txt --fout=pred.txt.out
prompt_6$ head -2 pred.txt.out
0.000,/RelValQCDForPF_14TeV/CMSSW_6_2_0_SLHC22_patch1-PH2_1K_FB_V6_UPG23SHNoTaper-v1/GEN-SIM-DIGI-RAW
0.000,/RelValQCDForPF_14TeV/CMSSW_6_2_0_SLHC22_patch1-PH2_1K_FB_V6_UPG23SHNoTaper-v1/DQMIO

# get popular datasets from popularity DB
prompt_7$ popular_datasets --start=20150101 --stop=20150108 > popdb-20150101-20150108.txt

# verify our prediction against similar period from popularity DB
prompt_8$ verify_predictions --pred=pred.txt.out --popdb=popdb-20150101-20150108.txt
Popular datasets   : 841
Predicted datasets : 187
Wrongly predicted  : 0
```

DISCUSSION

- Shown steps demonstrate ability of DCAFPilot project
- The results should be taken with caution
  - New data corresponded to first week of the year when there were no "real" activity among physicists
  - Chosen naccess metric may have bias towards test datasets which should be discarded
  - We may need to scan metric space for suitable definition of dataset "popularity"
- Use rolling approach: get new data → adjust model → make prediction and repeat the cycle
- We may need to extend existing dataframe to new dimensions: cluster users activity via HN analysis, conference dates; analysis of release quality, etc.

PRELIMINARY RESULTS

Following table shows result from model trained on Jan-Nov data and validated with Dec dataset (a la rolling approach). The RF, SGD, LinearSVC are scikit-learn classifiers (python), the Vowpal wabbit is online-learning algorithm by Yahoo, while eXtreme Gradient Boosting is parallel gradient boosting tree solution which won Kaggle Higgs competition.

		naccess>100	naccess>0	naccess>10 and naccess<10000 and nsites<50
Classifier	Data	accu prec reca f1	accu prec reca f1	accu prec reca f1
Random Forest	all	0.97 0.85 0.89 0.87	0.89 0.00 0.00 0.00	0.96 0.70 0.89 0.78
	new	0.83 1.00 0.83 0.91	0.00 0.00 0.00 0.00	0.92 1.00 0.92 0.96
SGDClassifier	all	0.97 0.88 0.68 0.77	0.95 0.86 0.70 0.77	0.95 0.70 0.72 0.71
	new	0.45 1.00 0.45 0.62	0.71 1.00 0.71 0.83	0.60 1.00 0.60 0.75
Linear SVC	all	0.94 0.53 0.99 0.69	0.97 0.83 0.97 0.89	0.95 0.62 0.92 0.74
	new	0.98 1.00 0.98 0.99	0.97 1.00 0.97 0.98	0.90 1.00 0.90 0.95
Vowpal Wabbit	all	0.95 0.61 0.69 0.65	0.99 0.91 1.00 0.95	0.94 0.65 0.61 0.63
	new	0.54 1.00 0.54 0.70	1.00 1.00 1.00 1.00	0.49 1.00 0.49 0.65
xgboost	all	0.98 0.88 0.95 0.92	0.99 0.97 1.00 0.98	0.96 0.71 0.97 0.82
	new	0.92 1.00 0.92 0.95	1.00 1.00 1.00 1.00	0.98 1.00 0.98 0.99

Data selection: rows with all values are data from train (Jan-Nov)/validation (Dec) sets, rows with new values are selected new datasets in Dec dataset, i.e. they were not present in train set.

CONCLUSIONS & FUTURE DIRECTIONS

- We show the proof of concept how to predict dataset popularity based on existing CMS tools
  - DCAFPilot package has main components to do the work, but does not limit you to use other tools
- We succeed making sensible prediction with different ML models
  - Even though initial dataframe/model shows some potential it should be thoughtfully studied to avoid main ML obstacles, e.g. data memorization, over-fitting, etc., and checked with new data
  - More data in terms of volume and attributes may be required for further analysis, e.g. find physicists clustering on certain topics
  - Even though all work was done on a single node with existing APIs we may need to pursue other approaches, e.g. ORACLE-Hadoop mapping, etc.
- Explore various ML algorithms: python, R, online-learning
- Try out different popularity metrics, e.g. (r)naccess, (r)totcpu, (r)nusers or any combination of them
- Explore different approaches: track individual datasets, dataset groups, etc.
- Use other resources: user activity on HN, conference deadlines influence, etc.
- Test predictions with real data, i.e. acquire new datasets and make prediction for them, then wait for data from popularity DB and compare prediction with actual data
- Automate tools, e.g. weekly crontabs, generate model updates, verify model predictions