Сравнение качества предсказательных моделей

Никита Казеев[1], Андрей Устюжанин[1,2]

[1]Yandex School of Data Analysis, [2]National Research University Higher School of Economics

Contents

- kaggle scripts, notebooks
- sacred
- rep
- openml
- git + CI + dashboard

kaggle scripts, notebooks

- Pros
  - easy to use
  - easy to fork
- Cons
  - computational limits at kaggle
  - difficult to run locally
  - difficult to migrate from local to kaggle
  - tied to Kaggle

sacred https://github.com/IDSIA/sacred

```
C = 1.0
gamma = 0.7

iris = datasets.load_iris()
perm = permutation(iris.target.size)
iris.data = iris.data[perm]
iris.target = iris.target[perm]
clf = svm.SVC(C, 'rbf', gamma=gamma)
clf.fit(iris.data[:90], iris.target[:90])
print(clf.score(iris.data[90:], iris.target[90:]))
```

sacred

```
from sacred import Experiment
ex = Experiment('iris_rbf_svm')
@ex.config
def cfg():
  C = 1.0
  gamma = 0.7
@ex.automain
def run(C, gamma):
  iris = datasets.load_iris()
  per = permutation(iris.target.size)
  iris.data = iris.data[per]
  iris.target = iris.target[per]
  clf = svm.SVC(C, 'rbf', gamma=gamma)
  ...
```

Running

```
# run mongoDB with DB collection
python experiment1-sacred.py with C=5.0
python experiment1-sacred.py -m DB
```

Reading results

- console example (see `src`)

```
$ ./read_sacred.py
                                result    C        seed
_id
571bf77016119c615e486650  0.983333  5.0  311911620
571bf1f016119c5ca3430d62  0.950000  1.0  252449081
571c847616119c15dd605002  0.950000  1.0  369063048
```

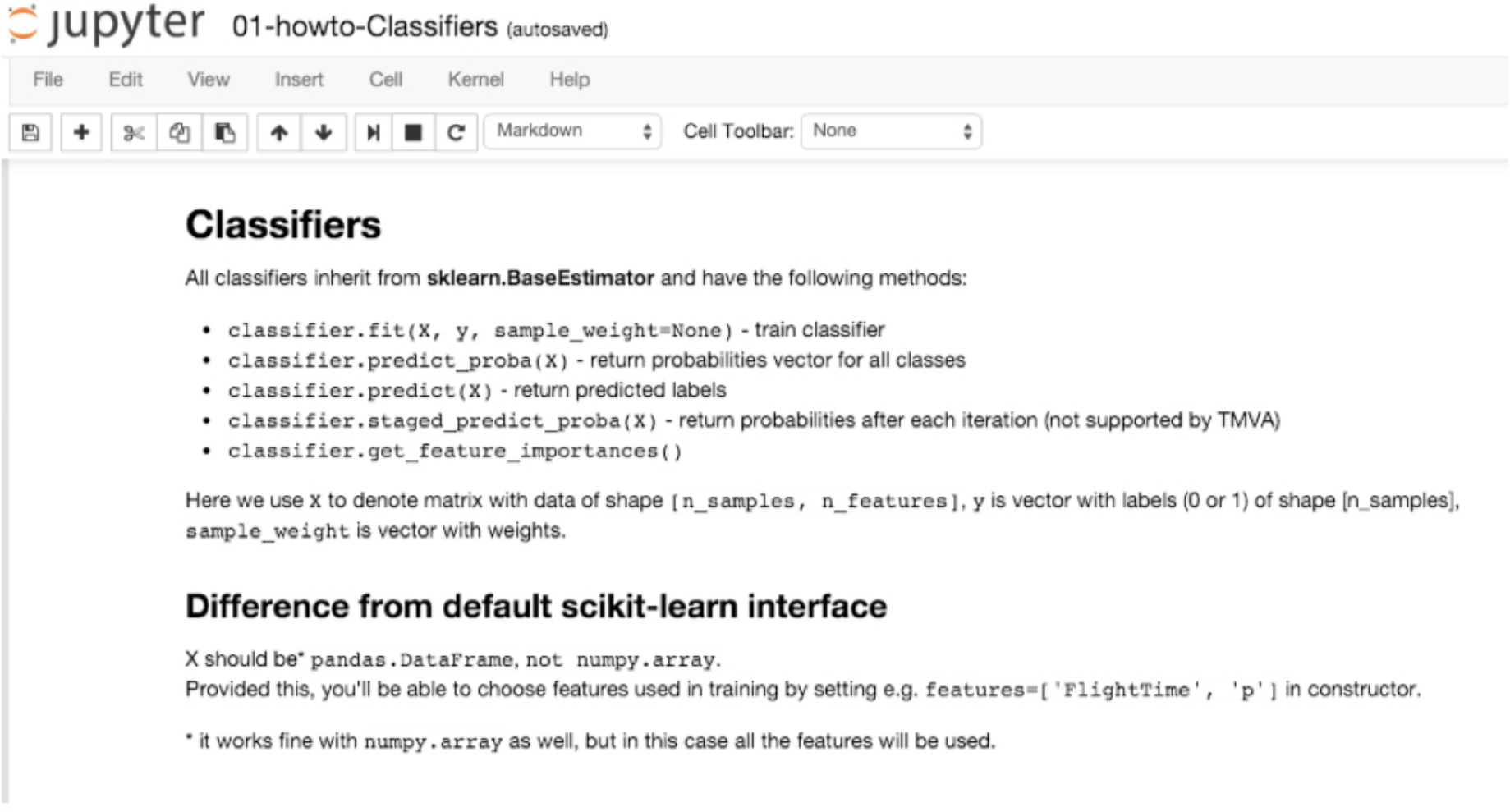
- not-so-ready-frontend
  - `https://github.com/Qwlouse/prophet`
  - `http://restheart.org/`

Reproducible Experiment Platform

- Python-based (numpy, pandas, ...), Jupyter-friendly
- Unified scikit-learn-like API to many ML packages (Sklearn, XGBoost, uBoost, TMVA, Theanets, ...)
- Meta-algorithms pipelines («REP lego»)
- Configurable interactive reporting & visualization to ensure model quality (e.g. check for overfitting)
- Pluggable quality metrics
- Parallelized training of classifiers & grid search (IPython parallel)
- Open-sourced, Apache 2.0:
  `https://github.com/yandex/rep`
- Supported by Yandex

Unified Classifier Interface

`https://github.com/yandex/rep/blob/master/howto/01-howto-Classifiers.ipynb`



Meta Machine Learning (REL-Lego)

- Factory
- Grid Search
  - GridOptimalSearch
  - Folding Scorer
  - Various Optimization algorithms
- Interface of parameter optimizer
- Folding `https://github.com/yandex/rep/blob/master/howto/04-howto-folding.ipynb`
- Stacking

REP-Lego

https://github.com/yandex/rep/blob/master/howto/01-howto-Classifiers.ipynb

jupyter 01-howto-Classifiers (autosaved)

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: None

AdaBoost over XGBoost

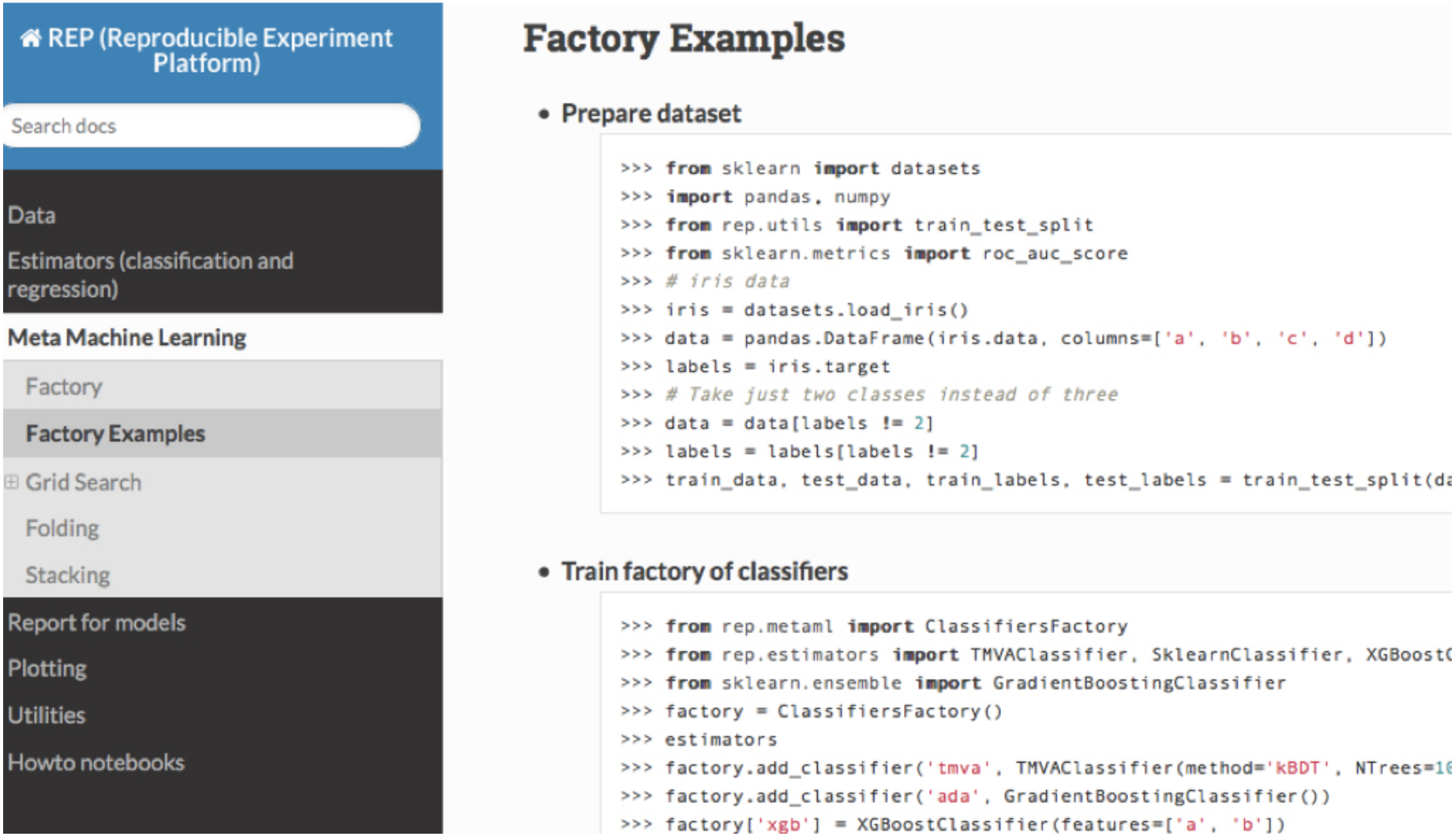
In [21]:
```
from sklearn.ensemble import AdaBoostClassifier
```

In [22]:
```
%%time
base_xgb = XGBoostClassifier(n_estimators=20)
ada_xgb = SklearnClassifier(AdaBoostClassifier(base_estimator=base_xgb, n_estimators=5))
ada_xgb.fit(train_data[variables], train_labels)
print('training complete!')

# predict probabilities for each class
prob = ada_xgb.predict_proba(test_data[variables])
print 'AUC', roc_auc_score(test_labels, prob[:, 1])

# predict probabilities for each class
prob = ada_xgb.predict_proba(train_data[variables])
print 'AUC', roc_auc_score(train_labels, prob[:, 1])
```

```
training complete!
AUC 0.975709190087
AUC 0.998466758443
CPU times: user 34 s, sys: 266 ms, total: 34.3 s
Wall time: 34.4 s
```

REP-Lego. Classifier Factories

https://github.com/yandex/rep/blob/master/howto/02-howto-Factory.ipynb



Reporting

- Draws set of reports upon model training completion. Supported libraries:
  - Matplotlib
  - ROOT
  - Bokeh (Javascript)
  - plot.ly (going to be deprecated due to limitations)
- Extensible!

`https://github.com/yandex/rep/blob/master/howto/02-howto-Factory.ipynb`

Reporting

jupyter 02-howto-Factory (autosaved)

File Edit View Insert Cell Kernel Help Python 2

Markdown Cell Toolbar: None

Get ClassificationReport object

report has many useful methods

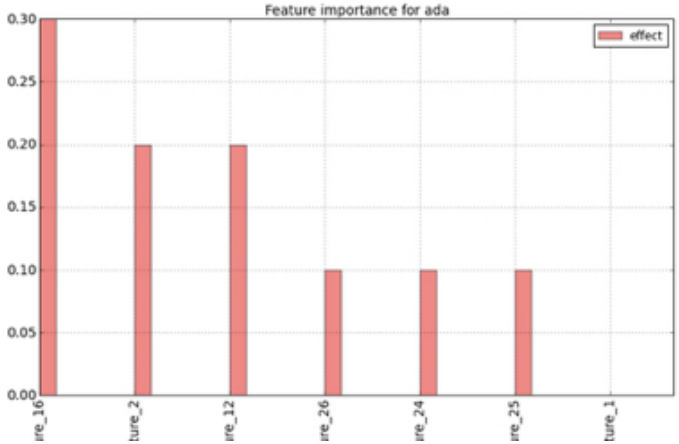
In [12]:
```
report = factory.test_on(test_data, test_labels)
```

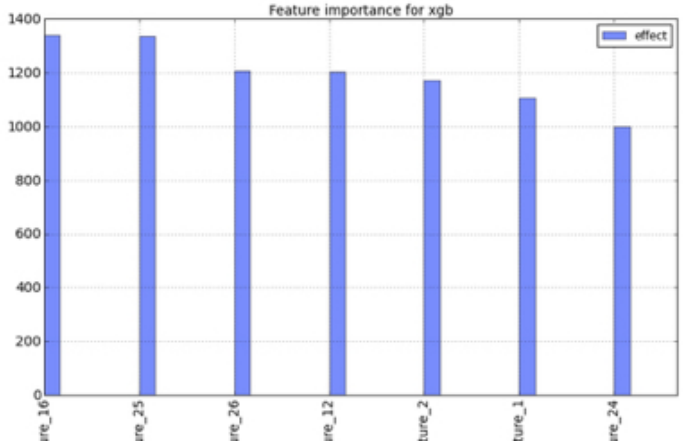
Plot importances of features

Only the features used in training are compared

In [13]:
```
features_importances = report.feature_importance()
features_importances.plot()
# not only in matplotlib, but in other libraries too. For instance, with plotly
# features_importances.plot_plotly('importances', figsize=(15, 6))
```

```
Estimator tmva doesn't support feature importances
```





More plots

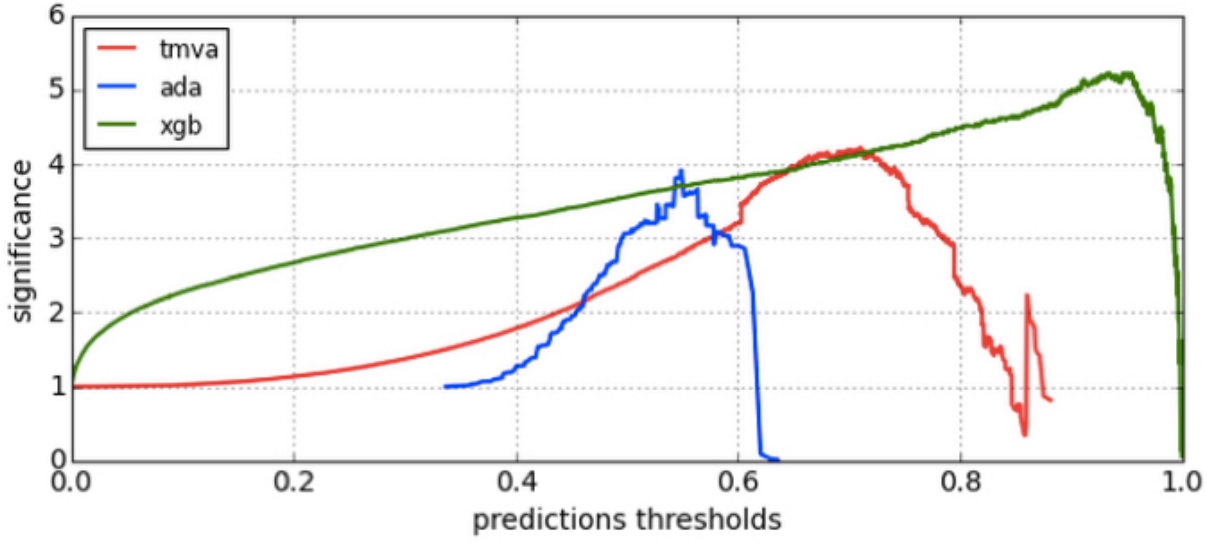

FunctionsPlot
matplotlib
matplotlib using mpld3 interactive plots
plotly interactive plots
bokeh interactive plots
ColorMap
Plot correlation between features
Plot data information: features correlation matrix
ROC curves (receiver operating characteristic)
Plot 'flatness' of classifier prediction
Quality as different metrics

Quality Metrics

Quality on different metrics

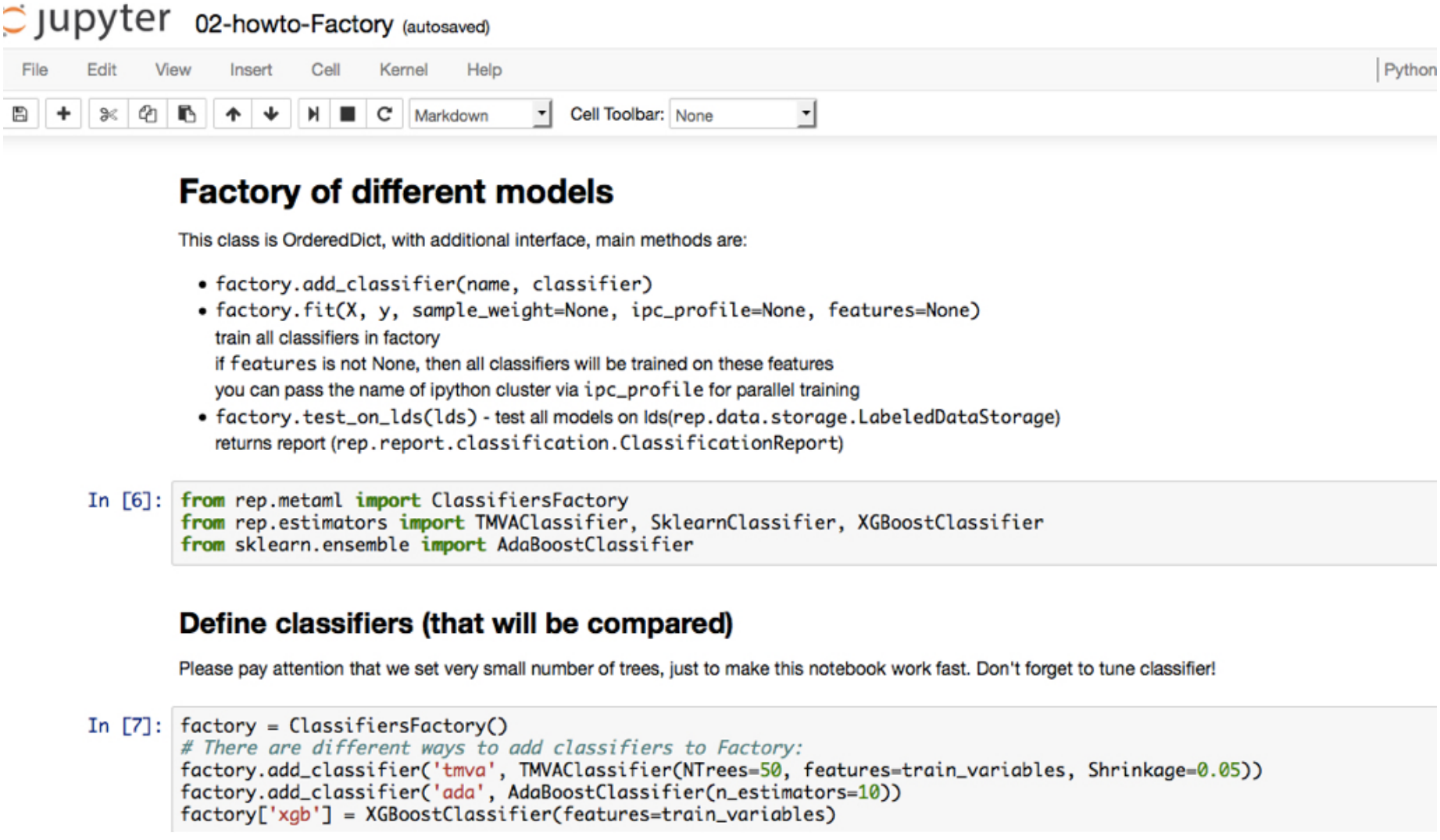
look how you can estimate the quality with your custom binary metrics and look for optimal threshold

In [24]:
```
from rep.report.metrics import significance
metrics = report.metrics_vs_cut(significance, metric_label='significance')
metrics.plot(new_plot=True, figsize=(10, 4))

# You can define your own metric and use it

# def AMS(s, b):
#     b_reg = 0.01
#     radicand = 2 *( (s+b+b_reg) * numpy.log (1.0 + s/(b+b_reg)) - s)
#     return numpy.sqrt(radicand)

# metrics = report.metrics_vs_cut(AMS, metric_label='ams')
```



Parallelized training & optimization

https://github.com/yandex/rep/blob/master/howto/02-howto-Factory.ipynb



Parallelized optimization. Grid Search

https://github.com/yandex/rep/blob/master/howto/02-howto-gridsearch.ipynb

In [4]:
```
import numpy
import numexpr
import pandas
from rep import utils
from sklearn.ensemble import GradientBoostingClassifier
from rep.report.metrics import RocAuc
from rep.metaml import GridOptimalSearchCV, FoldingScorer, RandomParameterOptimizer
from rep.estimators import SklearnClassifier, TMVAClassifier, XGBoostRegressor
```

In [5]:
```
# define grid parameters
grid_param = {}
grid_param['learning_rate'] = [0.2, 0.1, 0.05, 0.02, 0.01]
grid_param['max_depth'] = [2, 3, 4, 5]

# use random hyperparameter optimization algorithm
generator = RandomParameterOptimizer(grid_param)

# define folding scorer
scorer = FoldingScorer(RocAuc(), folds=3, fold_checks=3)
```

In [6]:
```
%%time
estimator = SklearnClassifier(GradientBoostingClassifier(n_estimators=30))
grid_finder = GridOptimalSearchCV(estimator, generator, scorer, parallel_profile='threads-4')
grid_finder.fit(data, labels)
```

```
Performing grid search in 4 threads
4 evaluations done
8 evaluations done
10 evaluations done
CPU times: user 47.2 s, sys: 772 ms, total: 48 s
Wall time: 17.5 s
```

Running REP

- Locally (virtualenv, conda)
  - `https://github.com/yandex/rep/wiki/Installing-manually`
- Locally (docker, kitematic)
  - `https://github.com/yandex/rep/wiki/`
- openstack (CERN) or any cloud provider
  - `https://github.com/yandex/rep/wiki/Install-REP-at-openstack`

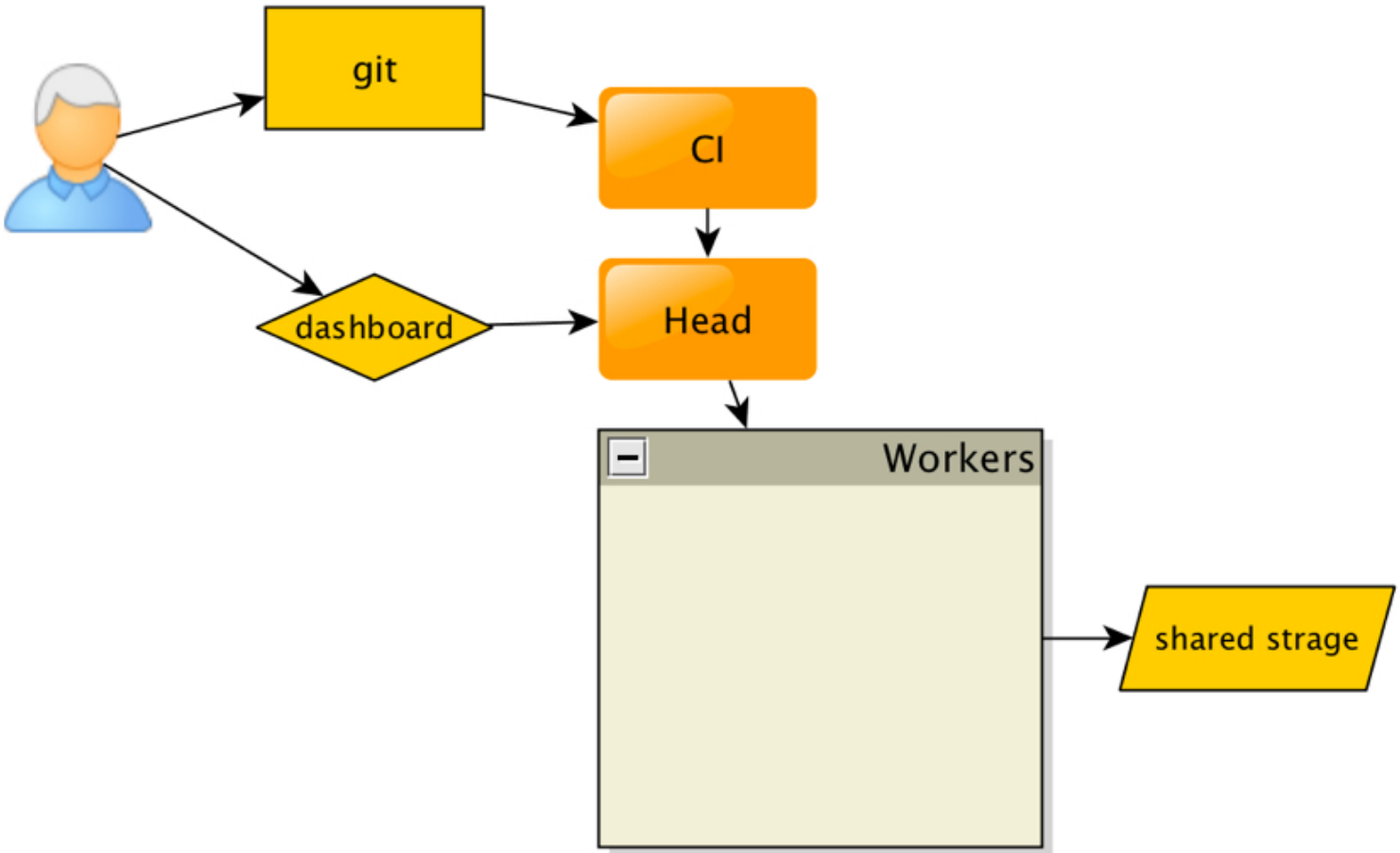
REP for research and eductaion

- High Energy Physics
  - online & offline data analysis at CERN
  - optimization of disk storage
- AstroPhysics
  - CRAYFIS (`http://crayfis.io`)
- Industry
  - Yandex Data Factory
- Education
  - MLHEP summer school (`http://www.hse.ru/mlhep2015/`)
  - Several hackathons (e.g. `https://cs.hse.ru/datanight`)

OpenML

- demo

Proper ML algorithm comparison



Conclusion

- comparison is not easy
  - kaggle
  - OpenML
- some tools might be helpful
  - sacred
  - rep
- to compare properly, no out-of-the-box solution is available, AFAWK

Спасибо за внимание!