Automatic Machine Learning (AutoML)

Using Python and Scikit-Learn

Abhishek Thakur

Paris 2016

Introduction

- Are industries really concerned with best algorithms and best results?
- 2-3% improvement in accuracy?

Introduction

- Are industries really concerned with best algorithms and best results?
- 2-3% improvement in accuracy?
- AutoML = One-Click Machine Learning
- Predictive models without looking at data
- Who is doing AutoML:
    - Google Predict
    - BigML
    - Dataiku
    - DataRobot
    - Me
    - etc. etc.

The AutoML Challenge

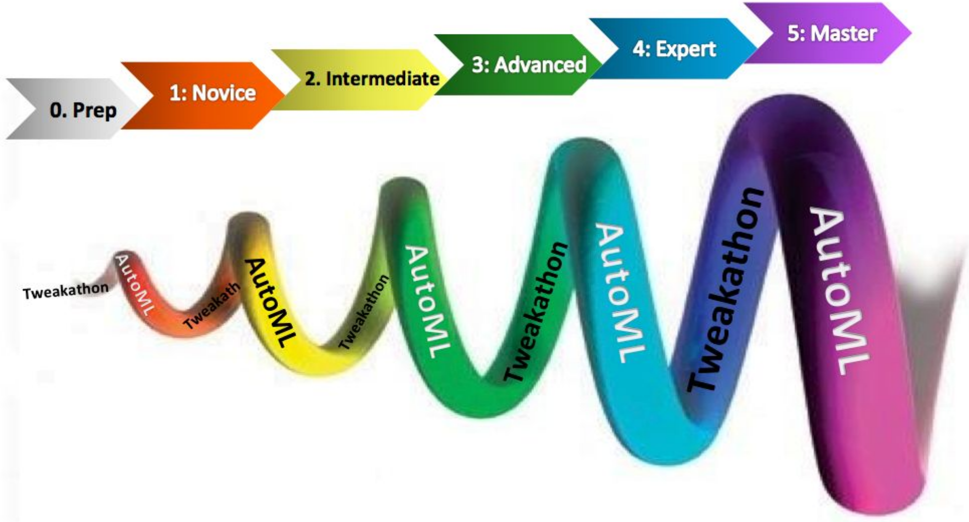


Image Source: http://www.causality.inf.ethz.ch/AutoML/spiral.png

https://competitions.codalab.org/competitions/2321

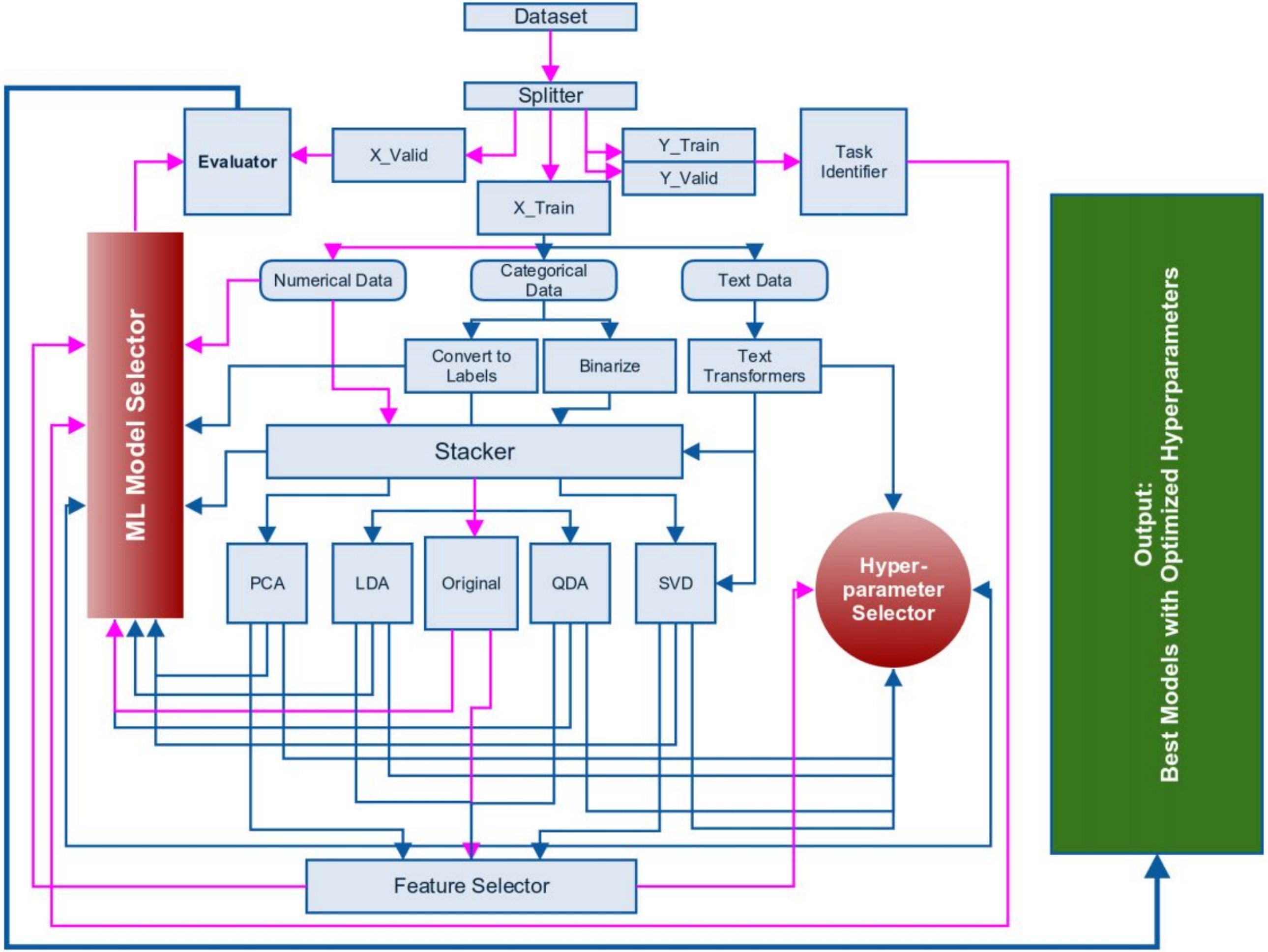
The AutoML Challenge

- Large scale evaluation of fully automatic learning machines
- 30 different datasets
- Over 5 rounds
- Lasted for ~1.5 years
- http://codalab.org/AutoML

AutoCompete

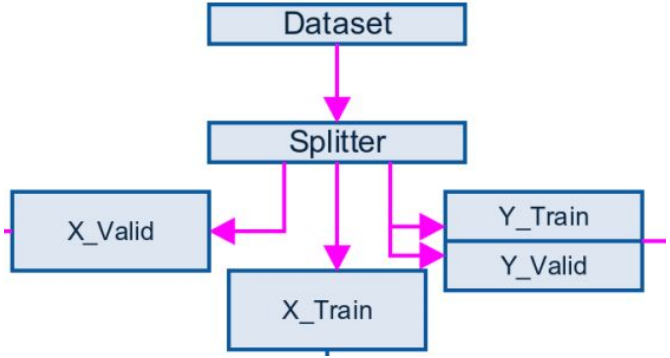
- Automated machine learning framework
- Tackles ML competitions
- Pipeline includes
  - Stratified data splitting
  - Feature building
  - Feature selection
  - Model and hyperparameter selection
  - Ensembler
- Search space: prior knowledge on different datasets
- Faster and comparable results compared to current methods
- Python + scikit-learn :)

Base Framework

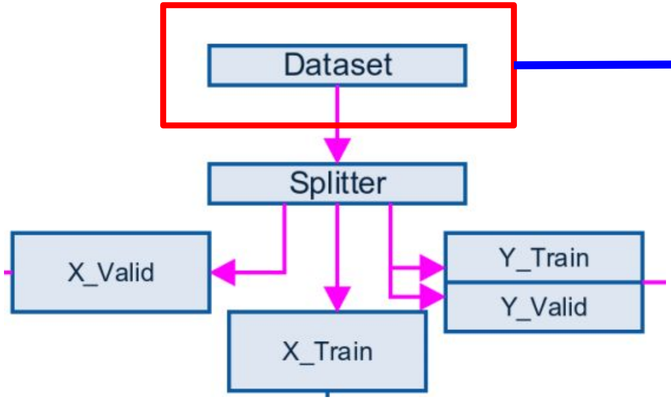


AutoCompete: A Framework for Machine Learning Competitions, A.Thakur and A Krohn-Grimberghe, ICML AutoML Workshop, 2015

Base Framework

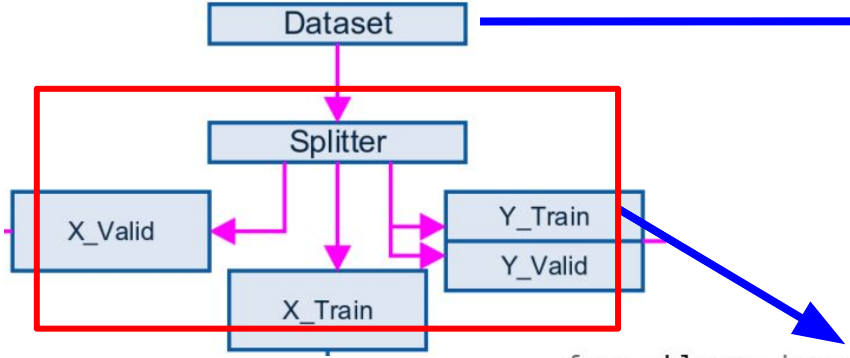


Base Framework




```python
import pandas as pd
df = pd.read_csv('data.csv')
```

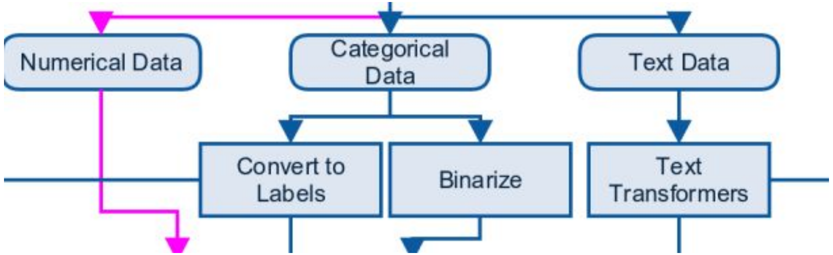
Base Framework



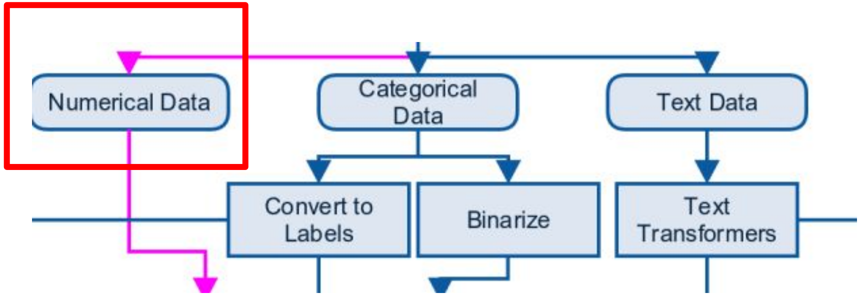

```python
import pandas as pd
df = pd.read_csv('data.csv')
```

```python
from sklearn import cross_validation

# for a classification problem
xtrain, xtest, ytrain, ytest = cross_valdation.train_test_split(X, y, stratify=y)

# for a regression problem
xtrain, xtest, ytrain, ytest = cross_valdation.train_test_split(X, y)
```

Base Framework

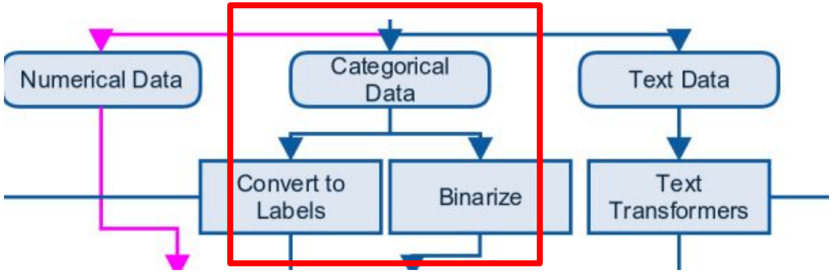


Base Framework



- Numerical Data:
  - Do nothing

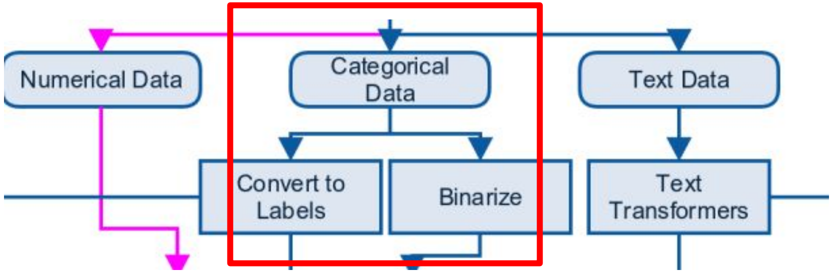
Base Framework



- Numerical Data:
    - Do nothing

- Categorical Data:
    - Label encoding
    - One-hot encoding

Base Framework

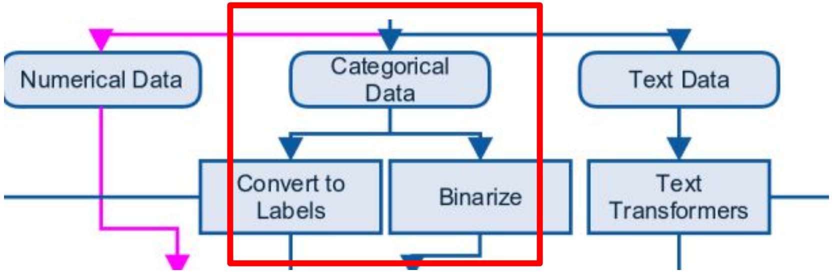


- Numerical Data:
  - Do nothing

- Categorical Data:
  - Label encoding
  - One-hot encoding

```
from sklearn.preprocessing import LabelEncoder

lbl_enc = LabelEncoder()
lbl_enc.fit(xtrain[categorical_features])
xtrain_cat = lbl_enc.transform(xtrain[categorical_features])
```

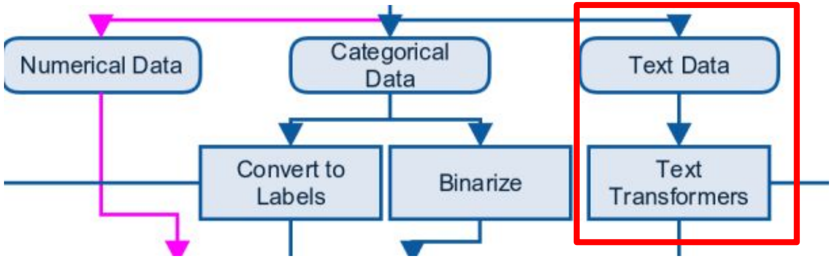
Base Framework



- Numerical Data:
  - Do nothing
- Categorical Data:
  - Label encoding
  - One-hot encoding

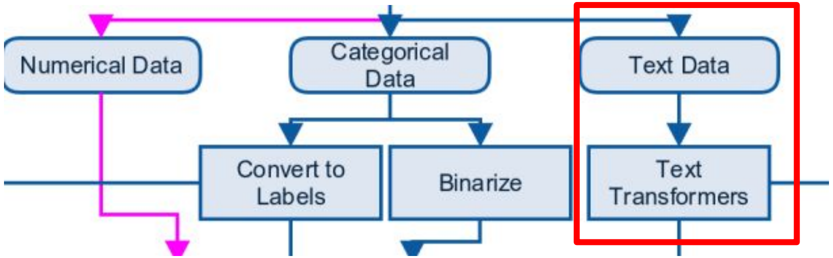
```
from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder()
ohe.fit(xtrain[categorical_features])
xtrain_cat = ohe.transform(xtrain[categorical_features])
```

Base Framework



- Numerical Data:
  - Do nothing
- Text Data:
  - Counts
  - TF-IDF

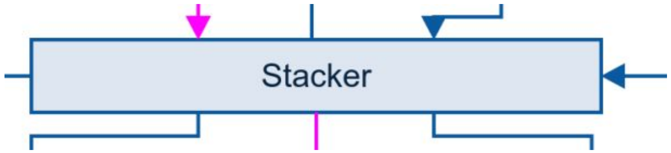
Base Framework



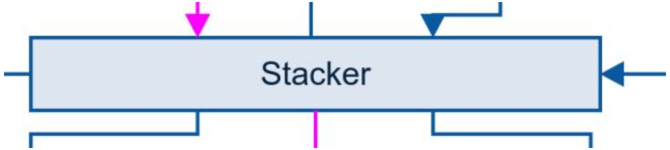
- Numerical Data:
  - Do nothing
- Text Data:
  - Counts
  - TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfv = TfidfVectorizer(min_df=3,  max_features=None,
        strip_accents='unicode', analyzer='word',token_pattern=r'\w{1,}',
        ngram_range=(1, 2), use_idf=1,smooth_idf=1,sublinear_tf=1,
        stop_words = 'english')
```

Base Framework

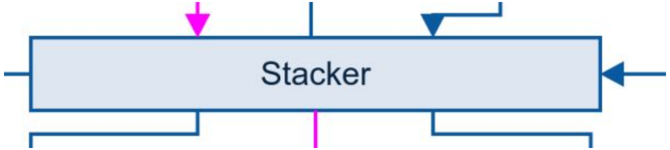


Base Framework




```python
import numpy as np
from scipy import sparse

# in case of dense data
X = np.hstack((x1, x2, ...))

# in case data is sparse
X = sparse.hstack((x1, x2, ...))
```

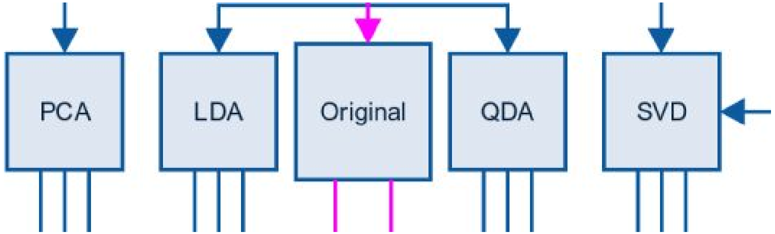
Base Framework



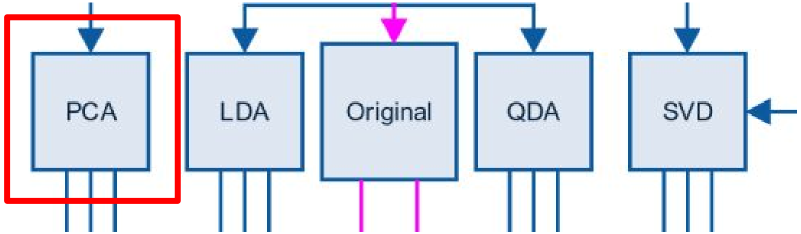

```
import numpy as np
from scipy import sparse

# in case of dense data
X = np.hstack((x1, x2, ...))

# in case data is sparse
X = sparse.hstack((x1, x2, ...))
```

```
from sklearn.pipeline import FeatureUnion
from sklearn.decomposition import PCA
from sklearn.feature_selection import SelectKBest

pca = PCA(n_components=10)
skb = SelectKBest(k=1)
combined_features = FeatureUnion([("pca", pca), ("skb", skb)])
```

Base Framework

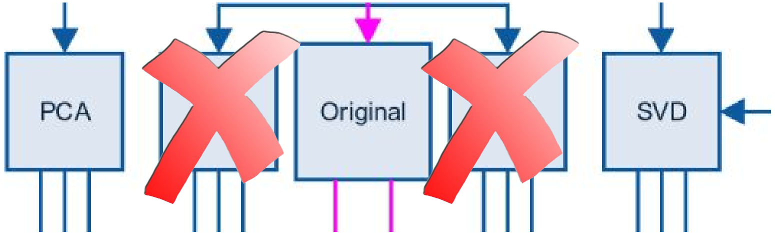


Base Framework

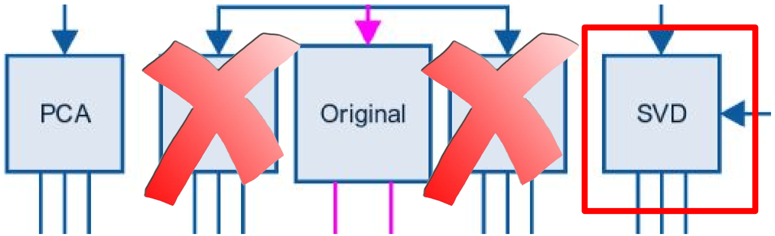



```
from sklearn.decomposition import PCA

pca = PCA(n_components=12)
pca.fit(xtrain)
xtrain = pca.transform(xtrain)
```

Base Framework

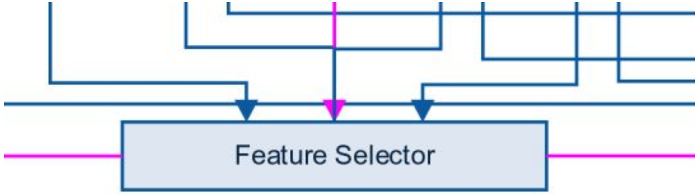


Base Framework



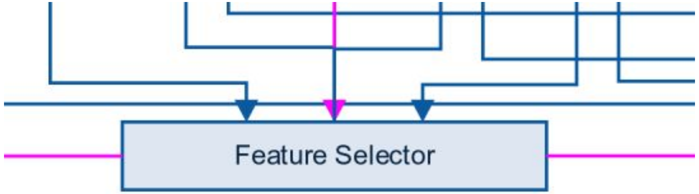

```
from sklearn.decomposition import TruncatedSVD

svd = TruncatedSVD(n_components=120)
svd.fit(xtrain)
xtrain = svd.transform(xtrain)
```

Base Framework



- Multiple ways of feature selection
- Random forest based feature importances
- Feature importances from GBM
- Chi2 feature selection
- Greedy feature selection

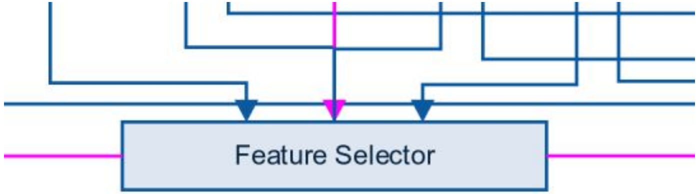
Base Framework



- Multiple ways of feature selection
- Random forest based feature importances
- Feature importances from GBM
- Chi2 feature selection
- Greedy feature selection

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=100, n_jobs=-1)
clf.fit(X, y)
X_selected = clf.transform(X)
```

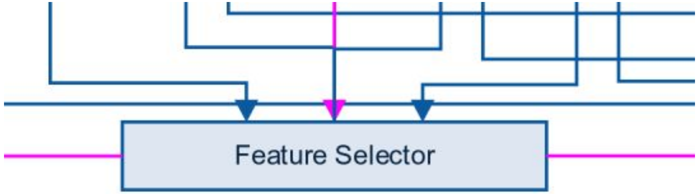
Base Framework



- Multiple ways of feature selection
- Random forest based feature importances
- Feature importances from GBM
- Chi2 feature selection
- Greedy feature selection

```
import xgboost as xgb

params = {}

model = xgb.train(params, dtrain, num_boost_round=100)
sorted(model.get_fscore().items(), key=lambda t: -t[1])
```

Base Framework



- Multiple ways of feature selection
- Random forest based feature importances
- Feature importances from GBM
- Chi2 feature selection
- Greedy feature selection

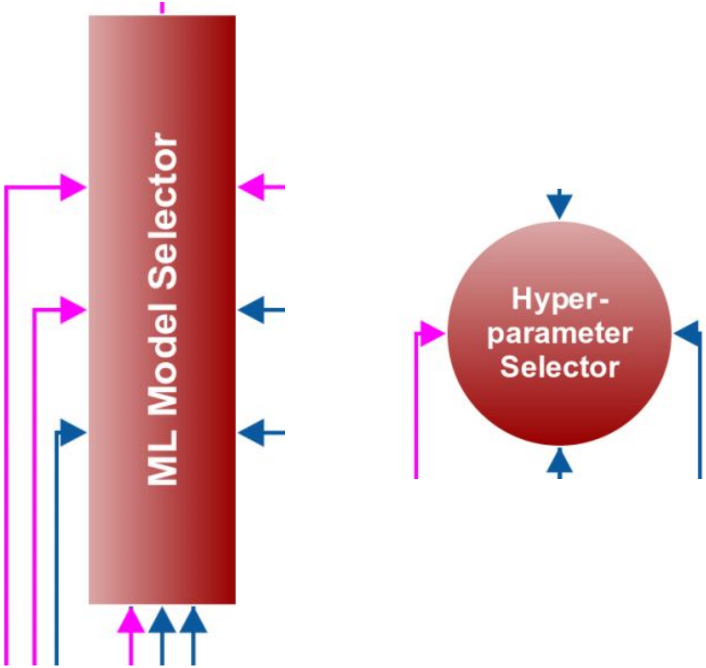
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

skb = SelectKBest(chi2, k=20)
skb.fit_transform(X, y)
```

Base Framework

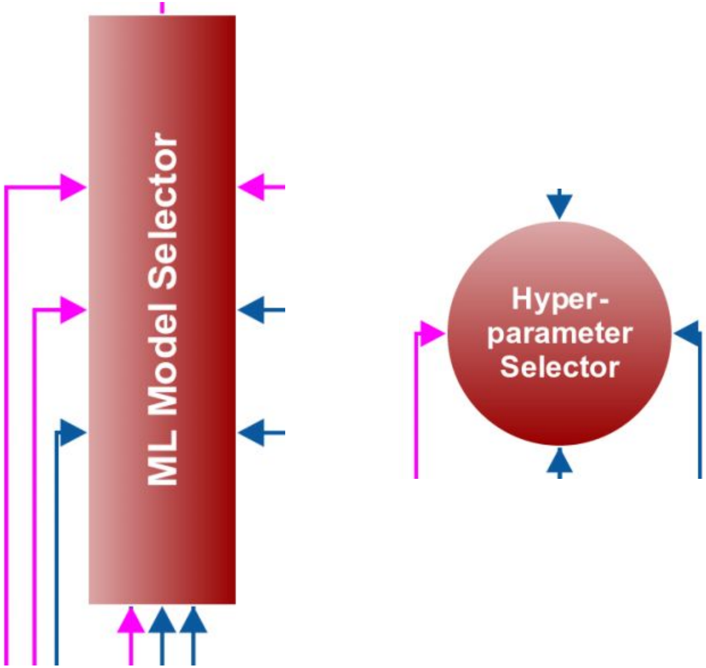
```
def selectionLoop(self, X, y):
        score_history = []
        good_features = set([])
        num_features = X.shape[1]
        while len(score_history) < 2 or score_history[-1][0] > score_history[-2][0]:
            scores = []
            for feature in range(num_features):
                if feature not in good_features:
                    selected_features = list(good_features) + [feature]

                    Xts = np.column_stack(X[:, j] for j in selected_features)

                    score = self.evaluateScore(Xts, y)
                    scores.append((score, feature))

                    if self._verbose:
                        print "Current AUC : ", np.mean(score)

            good_features.add(sorted(scores)[-1][1])
            score_history.append(sorted(scores)[-1])
            if self._verbose:
                print "Current Features : ", sorted(list(good_features))

        # Remove last added feature
        good_features.remove(score_history[-1][1])
        good_features = sorted(list(good_features))
        if self._verbose:
            print "Selected Features : ", good_features
```

- Multiple ways of feature selection
- Random forest based feature importances
- Feature importances from GBM
- Chi2 feature selection
- Greedy feature selection

Base Framework



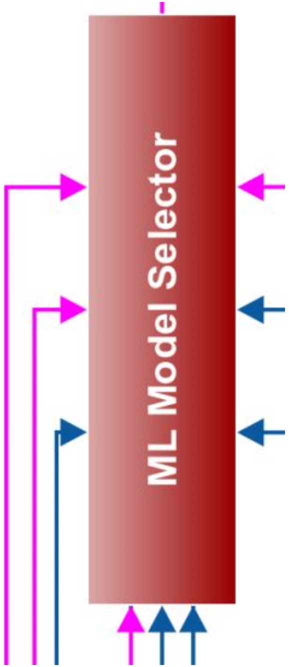
Base Framework

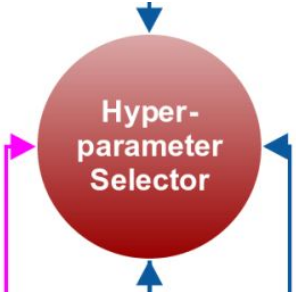


- Grid Search
- Random Search

Base Framework

- Classification:
    - Random Forest
    - GBM
    - Logistic Regression
    - Naive Bayes
    - Support Vector Machines
    - k-Nearest Neighbors



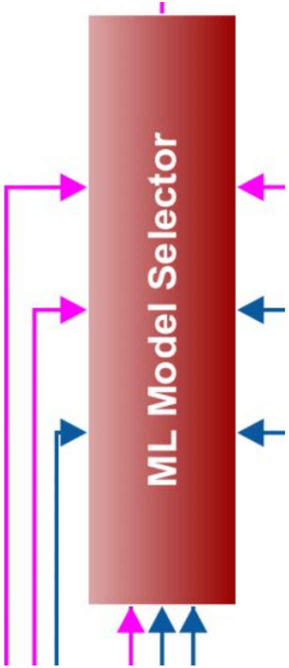


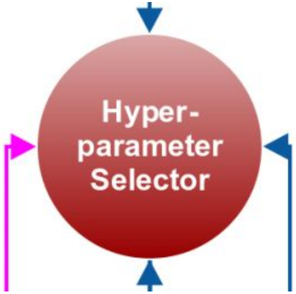
- Grid Search
- Random Search

Base Framework

- Classification:
    - Random Forest
    - GBM
    - Logistic Regression
    - Naive Bayes
    - Support Vector Machines
    - k-Nearest Neighbors

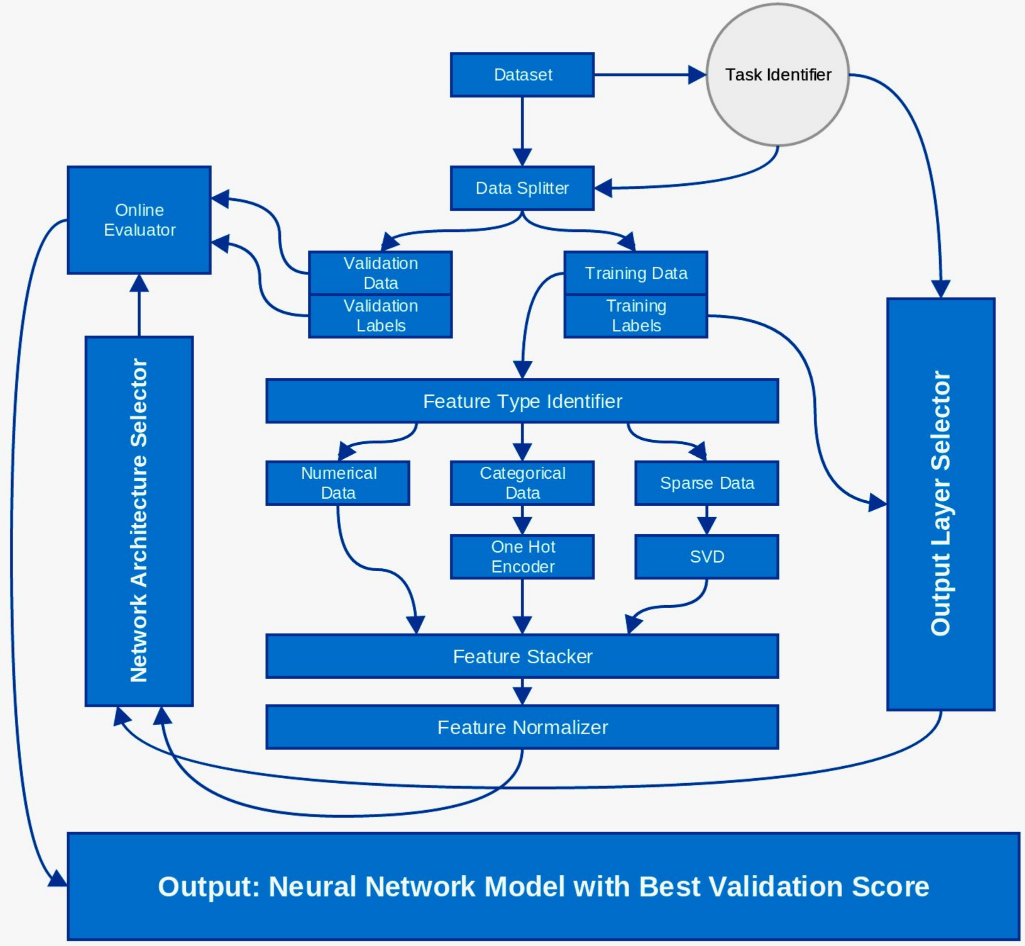
- Regression
    - Random Forest
    - GBM
    - Linear Regression
    - Ridge
    - Lasso
    - SVR





- Grid Search
- Random Search

A Similar Framework for Neural Nets

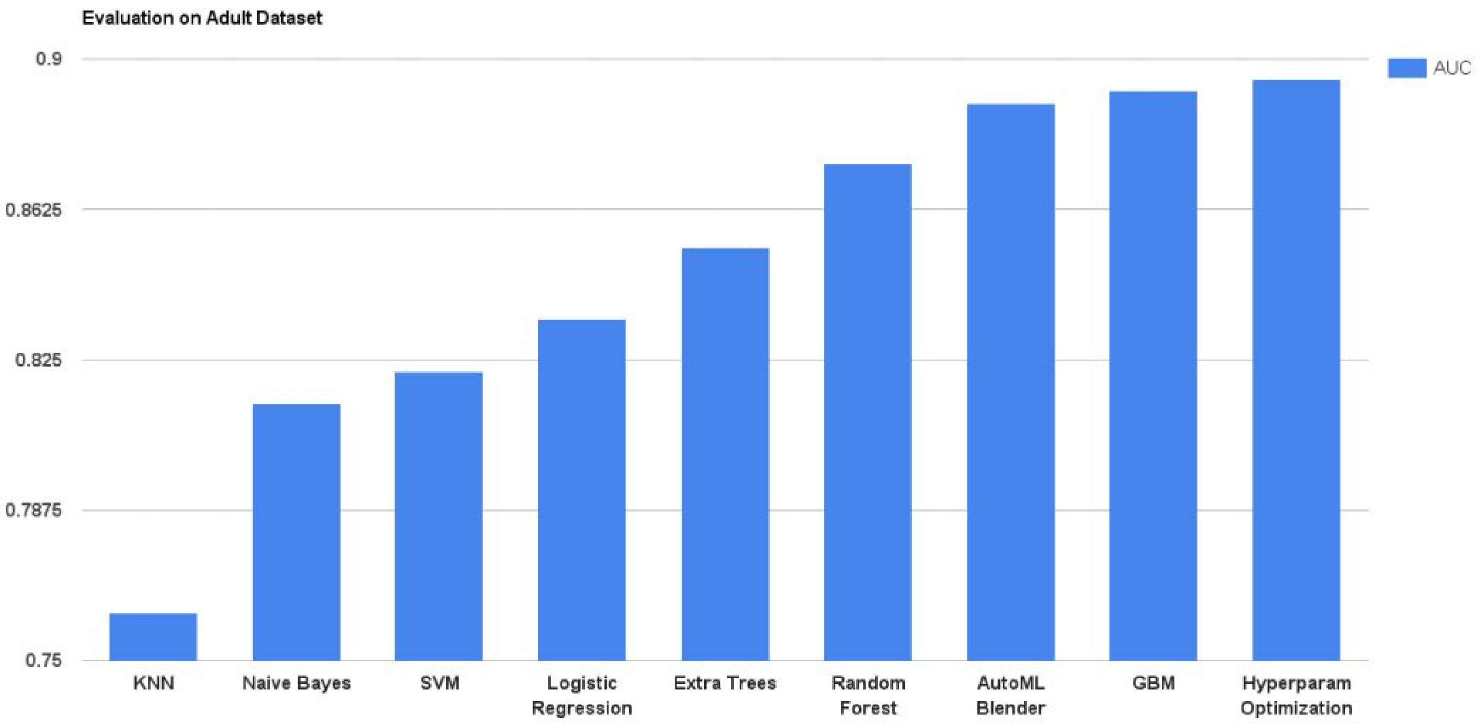


To Appear: AutoCompete 2.0: A Framework for Optimizing Parameters of Neural Networks, A.Thakur, ICML AutoML Workshop, System Desc Track, 2016

Selecting NNet Architecture

- Always use SGD or Adam (for fast convergence)
- Start low:
  - Single layer with 120-500 neurons
  - Batch normalization + PReLU
  - Dropout: 10-20%
- Add new layer:
  - 1200-1500 neurons
  - High dropout: 40-50%
- Very big network:
  - 8000-10000 neurons in each layer
  - 60-80% dropout

Some Experiments



Some Experiments

Algorithm	Weighted Average F1 Score
AutoCompete	**0.864**
hyperopt-sklearn	0.856
SVMTorch	0.848
LibSVM	0.843

Results on Newsgroups-20 dataset

Some Results

RESULTS							
	User	<Rank>	Set 1	Set 2	Set 3	Set 4	Set 5
1	ideal.intel.analytics	1.40 (1)	0.8262 (1)	0.8132 (2)	0.9632 (2)	0.8877 (1)	0.5894 (1)
2	abhishek4	3.60 (2)	0.8178 (4)	0.7924 (4)	0.9394 (5)	0.8716 (2)	0.4608 (3)
3	aad_freiburg	4.00 (3)	0.8172 (6)	0.8107 (3)	0.9751 (1)	0.8580 (5)	0.3958 (5)
4	asml.intel.com	7.40 (4)	0.8238 (3)	0.8172 (1)	0.9551 (3)	0.8484 (6)	0.3324 (24)
5	Rong	7.60 (5)	0.8136 (10)	0.7851 (7)	0.8740 (8)	0.8704 (3)	0.3367 (10)

AutoML Final1 Results

Some Results

RESULTS

	User	<Rank>	Set 1	Set 2	Set 3	Set 4	Set 5
1	aad_freiburg	1.60 (1)	0.6530 (1)	0.5175 (2)	0.2843 (1)	0.7821 (1)	0.3823 (3)
2	ideal.intel.analytics	3.60 (2)	0.6137 (3)	0.5263 (1)	0.2455 (5)	0.7271 (7)	0.3863 (2)
3	abhishek4	5.40 (3)	0.5946 (6)	0.5064 (6)	0.2251 (7)	0.7574 (3)	0.3720 (5)

AutoML Final4 Results

Some Results

Dataset	Evita	Flora	Helena	Tania	Yolanda
Evaluation Metric	AUC	A Metric	BAC	PAC	R2
Abhishek	0.5694	0.5001	0.2381	0.7617	0.3870
Damir	0.5816	0.5061	0.2469	0.7498	0.3654
AAD Frieburg	0.5866	0.4540	0.2673	0.7557	0.3782

AutoML GPU Track
Results

Conclusions

- We built a partially-automated framework to tackle tabular data
- The framework was extended to use neural networks
- The system gives results comparable to current best methods.
- Learning from past data instead of randomly choosing parameters in a fixed range, given a problem, makes the system faster than other systems.
- It is not fully-automated yet and work is in progress.
- The framework will be extended to auto-tuning of convolutional neural networks soon.

Comments / Questions

- @abhi1thakur
- bit.ly/thakurabhishek
- kaggle.com/abhishek