# Efficient and Robust Automated Machine Learning

Bucevschi Alexandru[1] and Stoleru Ingrid[1]

[1]Facultatea de Informatica
Universitatea "Alexandru Ioan Cuza"

January 16, 2020

# Outline

# AutoML

**Automatizarea procesului de aplicare a abordarilor de ML in problemele de zi cu zi.**

# AutoML - problemele adresate

- Detectia automata a tipurilor datelor de intrare
- Determinarea automata a task-ului: clusterizare, clasificare binara/ multi-class
- Automatizarea procesului de feature engineering (selection/extraction)
- Selectarea automata a modelului
- Optimizarea automata a hiperparametrilor

# Subiectul paper-ului

Paper-ul curent propune un nou sistem de **AutoML**, intitulat AUTO-SKLEARN.

Acesta este bazat pe **scikit-learn**:

- machine learning library (Python)
- 15 clasificatori
- 14 metode de feature preprocessing
- 4 metode de data preprocessing

# Avantajele sistemului propus

la in calcul in mod automat performantele anterioare pe dataset-uri similare si construieste ensemble-uri din modelele evaluate la faza de optimizare.

**Problema AutoML**:
Producerea automata (fara interventia utilizatorului) a predictiilor pentru un set nou de date, intr-un buget computational fix.

**Buget computational** = Memorie, CPU

**Definition 1** (AutoML problem). *For $i = 1, \ldots, n+m$, let $\boldsymbol{x}_i \in \mathbb{R}^d$ denote a feature vector and $y_i \in Y$ the corresponding target value. Given a training dataset $D_{train} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ and the feature vectors $\boldsymbol{x}_{n+1}, \ldots, \boldsymbol{x}_{n+m}$ of a test dataset $D_{test} = \{(\boldsymbol{x}_{n+1}, y_{n+1}), \ldots, (\boldsymbol{x}_{n+m}, y_{n+m})\}$ drawn from the same underlying data distribution, as well as a resource budget $b$ and a loss metric $\mathcal{L}(\cdot, \cdot)$, the AutoML problem is to (automatically) produce test set predictions $\hat{y}_{n+1}, \ldots, \hat{y}_{n+m}$. The loss of a solution $\hat{y}_{n+1}, \ldots, \hat{y}_{n+m}$ to the AutoML problem is given by $\frac{1}{m} \sum_{j=1}^{m} \mathcal{L}(\hat{y}_{n+j}, y_{n+j})$.*

# Probleme AutoML

- Niciun algoritm individual de ML nu performeaza foarte bine pe toate seturile de date.
- Unele abordari de ML se bazeaza pe optimizarea de hiperparametri (SVM-ul non-linear).

# Combined Algorithm Selection and Hyperparameter optimization

**Definition 2** (CASH). *Let $\mathcal{A} = \{A^{(1)}, \ldots, A^{(R)}\}$ be a set of algorithms, and let the hyperparameters of each algorithm $A^{(j)}$ have domain $\Lambda^{(j)}$. Further, let $D_{train} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training set which is split into $K$ cross-validation folds $\{D^{(1)}_{valid}, \ldots, D^{(K)}_{valid}\}$ and $\{D^{(1)}_{train}, \ldots, D^{(K)}_{train}\}$ such that $D^{(i)}_{train} = D_{train} \backslash D^{(i)}_{valid}$ for $i = 1, \ldots, K$. Finally, let $\mathcal{L}(A^{(j)}_{\boldsymbol{\lambda}}, D^{(i)}_{train}, D^{(i)}_{valid})$ denote the loss that algorithm $A^{(j)}$ achieves on $D^{(i)}_{valid}$ when trained on $D^{(i)}_{train}$ with hyperparameters $\boldsymbol{\lambda}$. Then, the* Combined Algorithm Selection and Hyperparameter optimization (CASH) *problem is to find the joint algorithm and hyperparameter setting that minimizes this loss:*

$$A^{\star}, \boldsymbol{\lambda}_{\star} \in \underset{A^{(j)} \in \mathcal{A}, \boldsymbol{\lambda} \in \Lambda^{(j)}}{\operatorname{argmin}} \frac{1}{K} \sum_{i=1}^{K} \mathcal{L}(A^{(j)}_{\boldsymbol{\lambda}}, D^{(i)}_{train}, D^{(i)}_{valid}). \tag{1}$$

# CASH - Aim

Incercam sa identificam o combinatie de algoritmi din tot spatiul de algoritmi, cu hiperparametrii corespunzatori, care performeaza cel mai bine pe toate seturile de date.
(Minimizeaza functia de loss, deci numarul de instante clasificate gresit)

# CASH - Aim

$$x^{\star} = \arg\min_{x \in \mathcal{X}} f(x)$$

# CASH - Blackblox

- Nu avem gradientii functiei
- Evaluarea functiei este costisitoare
- Optimizarea - Strategie secventiala (colectam date, apoi pe baza lor decidem ce punct vom evalua in continuare)

# Solutie

**Optimizarea Bayesiana**

- Metoda iterativa
- Tine cont de evaluarile din trecut, formand un model probabilistic
- Mapeaza hiperparametrii la probabilitatea de a obtine un anumit scor

# Optimizarea Bayesiana

$$P(\,score \mid hyperparameters)$$

# Optimizarea Bayesiana

- "Surogatul functiei obiectiv"
- Mai usor de optimizat decat functia obiectiv
- Selecteaza hiperparametrii care performeaza cel mai bine pe functia surogat
- Calculeaza valoarea functiei obiectiv si face update la surogat

# Optimizarea Bayesiana

**"Spend a little more time selecting the next hyperparameters in order to make fewer calls to the objective function."**

# Avantajele modelului propus

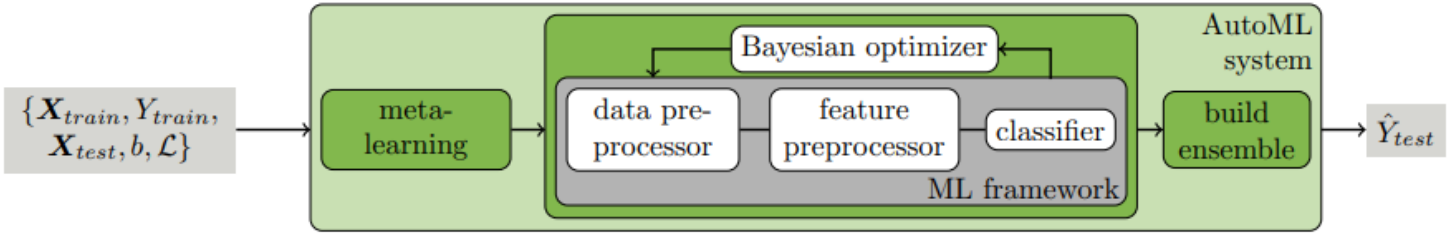



Figure 1: Our improved AutoML approach. We add two components to Bayesian hyperparameter optimization of an ML framework: meta-learning for initializing the Bayesian optimizer and automated ensemble construction from configurations evaluated during optimization.

# Complementaritatea

- Meta-learning-ul sugereaza instantieri ale framework-ului care sunt probabile sa performeze bine
- Optimizarea bayesiana realizeaza fine-tuningul pe partea de performanta

# Imaginea de ansamblu

- Se selecteaza k configurari pe baza meta-learningului
- Se initializeaza un algoritm de optimizare bayesiana cu aceste configurari

# Generarea configuratiilor initiale

- S-a pornit de la repo-ul de OpenML - **140 dataseturi**
- S-au implementat 38 de meta-feature-uri, ce caracterizeaza dataseturile
- Exemple de feature-uri:
  - Statistici despre numarul de instante, feature-uri si clase
  - Entropia label-urilor
  - Data skewness

# Meta Learning

- Pentru fiecare dataset, s-au evaluat feature-urile
- A fost folosita optimizarea bayesiana pentru a determina o instantiere a frameworkului cu rezultate foarte bune pe acel dataset.

# Evaluarea pe un dataset nou

- Se calculeaza meta-feature-urile
- Se calculeaza distanta L1 fata de celelalte dataset-uri in spatiul de feature-uri
- Se aleg cele mai apropiate 25 de dataset-uri si se stocheaza instantierile corespunzatoare, care sunt date optimizatorului

# Problema optimizarii bayesiene

O procedura ce consuma resurse utile, in sensul in care modelele care performeaza aproape la fel de bine ca cel mai bun sunt pierdute.

# Solutia

Stocheaza cele mai performante modele si printr-o serie de preprocesari, construieste un ansamblu din acestea.

- Robustete
- Reducerea overfitting-ului

- Weight-urile uniforme asociate modelelor individuale nu au condus la rezultate bune.
- Metoda optimizata de construire a ansamblurilor - **ensemble selection**.
- Pornim de la un empty ensemble si in mod iterativ adaugam modele care maximizeaza performanta ansamblului la validare.
- Modelele sunt adaugate cu weighturi uniforme, insa se permit repetitiile.
- Dimensiunea ansamblului: **50**

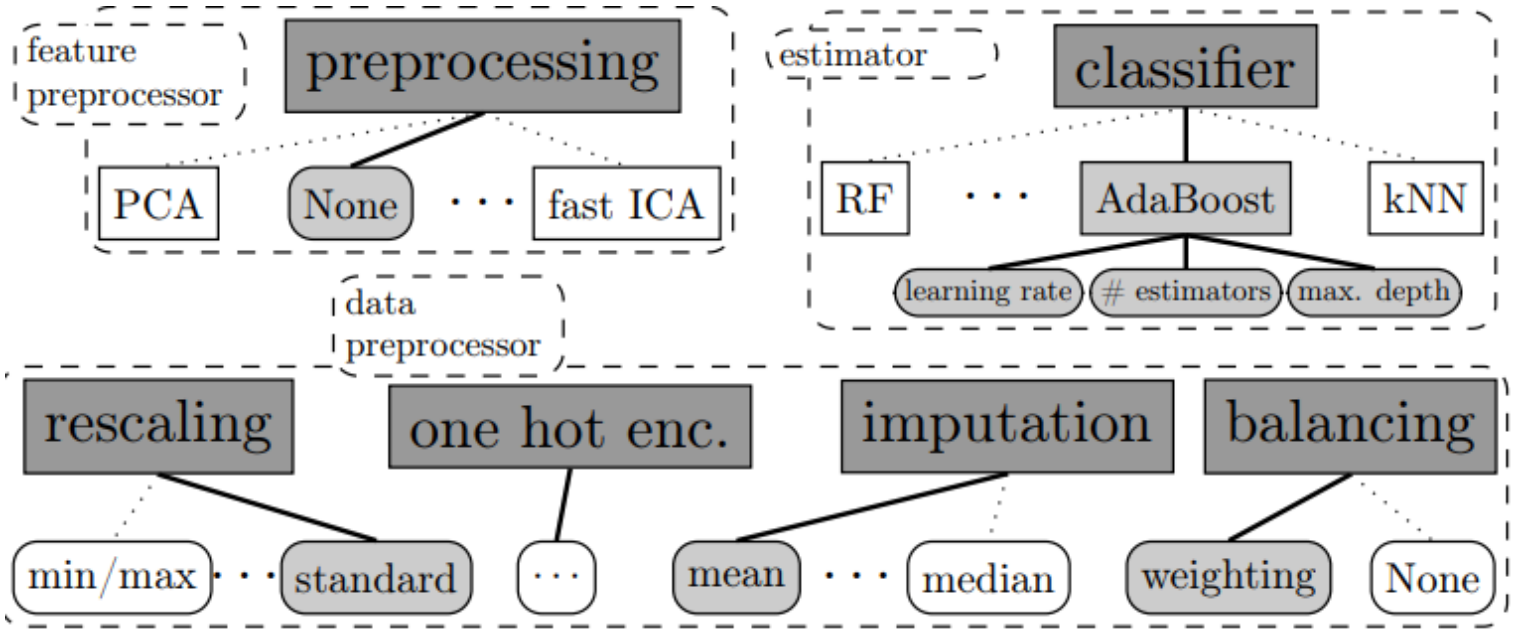
# Sisteme

- Porneste de la **scikit-learn** - ML library
- AUTO-WEKA (optimizarea bayesiana)
- AUTO-SKLEARN

# Exemple

- Bayes Naiv, Arbori de decizie, KNN, SVM, AdaBoost
- PCA, ICA, Polynomial
- One-hot encoding, balansare, rescalare

# Spatiul de configurare




THANK YOU FOR YOUR
ATTENTION