XVII Conferenza Italiana degli Utenti di Stata
19th-20th of May, 2022

Machine Learning using Stata/Python

Giovanni Cerulli

Consiglio Nazionale delle Ricerche
IRCrES
ISTITUTO di RICERCA sulla CRESCITA ECONOMICA SOSTENIBILE
INSTITUTE of RESEARCH on ECONOMIC SUSTAINABLE GROWTH

What is Machine Learning ?

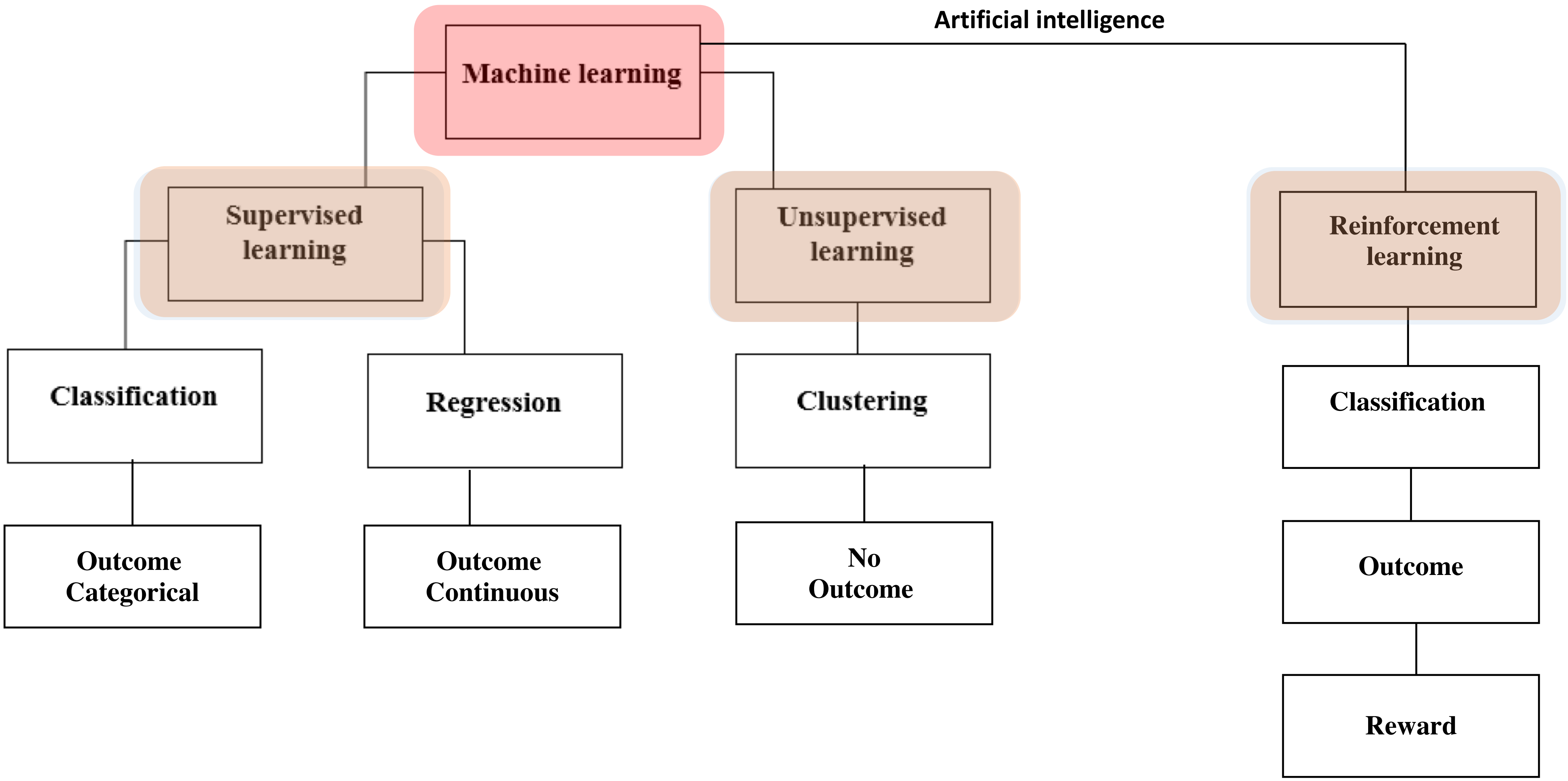
Machine Learning

A relatively new approach to **data analytics**, which places itself in the intersection between **statistics**, **computer science**, and **artificial intelligence**

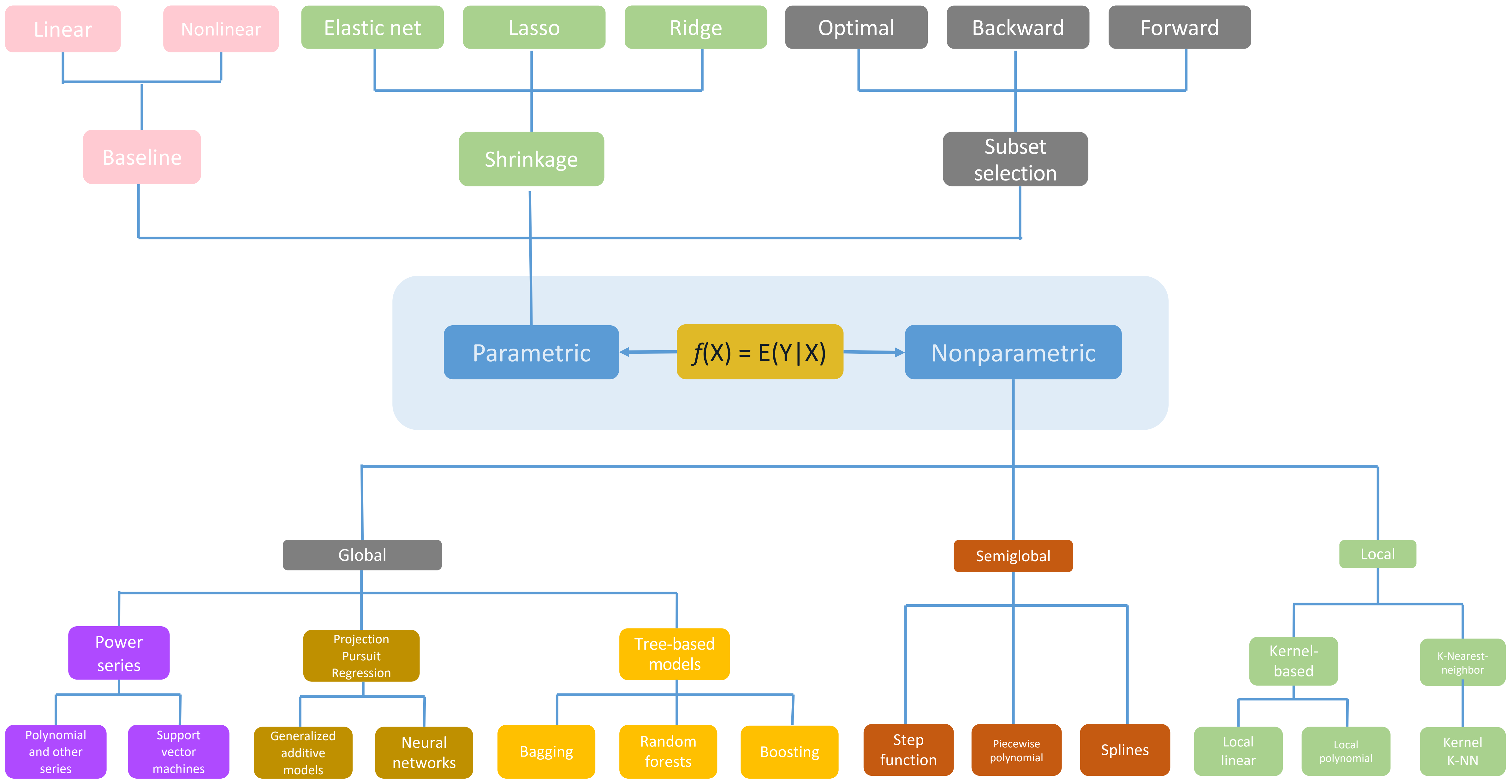
ML objective

Turning **information** into **knowledge** and **value** by **"letting the data speak"**

Supervised, Unsupervised, Reinforcement Learning



Supervised Machine Learning Methods



Hyper-parameter tuning

ML method	Parameter 1	Parameter 2	Parameter 3
Linear Models and GLM	N of covariates		
Lasso	Penalization coefficient		
Elastic-Net	Penalization coefficient	Elastic parameter	
Nearest-Neighbor	N of neighbors		
Neural Network	N of hidden layers	N of neurons	L2 penalization
Trees	N of leaves/depth		
Boosting	Learning parameter	N of sequential trees	N of leaves/depth
Random Forest	N of features for splitting	N of bootstraps	N of leaves/depth
Bagging	Tree-depth	N of bootstraps	
Support Vector Machine	C	Γ	
Kernel regression	Bandwidth	Kernel function	
Piecewise regression	N of knots		
Series regression	N of series terms		

Software for ML

Software

General purpose ML platform

Deep Learning platform

Deep Learning platform

scikit learn

TensorFlow

K Keras

Software

Python/Stata fully integrated platform via the SFI environment

Various ML packages but poor deep learning libraries

Statistics and Machine Learning Toolbox
Deep Learning Toolbox

Python Scikit-learn platform → `c_ml_stata_cv` & `r_ml_stata_cv` (Cerulli, 2022)

scikit-learn

Machine Learning in Python

Getting Started | Release Highlights for 0.24 | GitHub

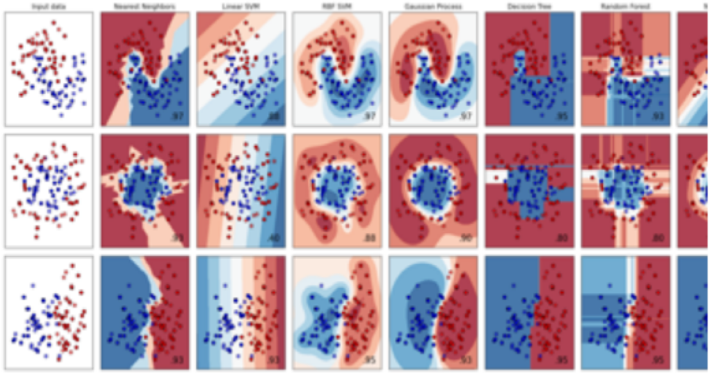
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



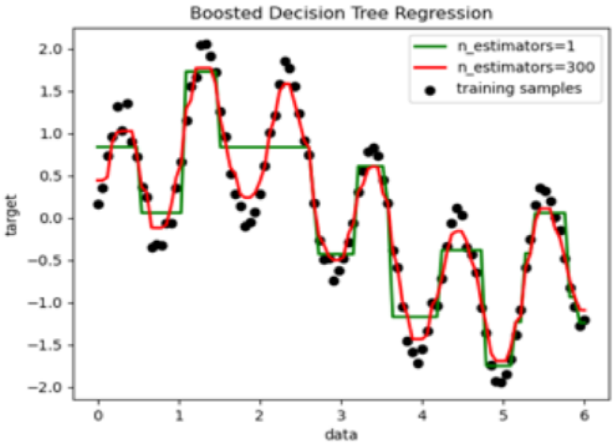
Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...



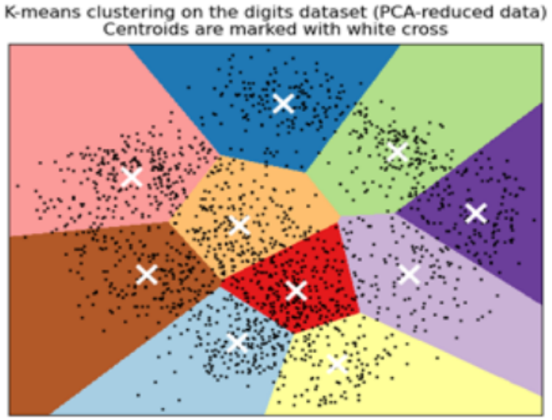
Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



Examples

Table of Contents

Next topic

Characteristic (sfi.Characteristic)

Quick search

Go

Stata's Python API documentation

The **Stata Function Interface (sfi)** module allows users to interact Python's capabilities with core features of Stata. The module can be used interactively or in do-files and ado-files.

Within the module, classes are defined to provide access to Stata's characteristics, current dataset, frames, date and time, macros, scalars, matrices, value labels, global Mata matrices, missing values, etc.

Class Summary

- Characteristic (sfi.Characteristic)
- Data (sfi.Data)
- Datetime (sfi.Datetime)
- Frame (sfi.Frame)
- Macro (sfi.Macro)
- Mata (sfi.Mata)
- Matrix (sfi.Matrix)
- Missing (sfi.Missing)
- Platform (sfi.Platform)
- Preference (sfi.Preference)
- Scalar (sfi.Scalar)
- SFIToolkit (sfi.SFIToolkit)
- StrLConnector (sfi.StrLConnector)
- ValueLabel (sfi.ValueLabel)

ML regression and classification with

`r_ml_stata_cv` & `c_ml_stata_cv`

Stata command r_ml_stata_cv

r_ml_stata_cv *depvar varlist* , mlmodel(*modeltype*) data_test(*filename*) seed(*integer*) [*learner_options cv_options other_options*]

modeltype_options	Description
Model	
ols	Ordinary least squares
elasticnet	Elastic net
tree	Tree regression
randomforest	Bagging and random forests
boost	Boosting
nearestneighbor	Nearest neighbor
neuralnet	Neural network
svm	Support vector machine

Regression

Stata command c_ml_stata_cv

c_ml_stata_cv *depvar varlist* , **mlmodel(***modeltype***) data_test(***filename***)** **seed(***integer***)** [*learner_options cv_options other_options*]

modeltype_options	Description
Model	
tree	Classification tree
randomforest	Bagging and random forests
boost	Boosting
regmult	Regularized multinomial
nearestneighbor	Nearest Neighbor
neuralnet	Neural network
naivebayes	Naive Bayes
svm	Support vector machine
multinomial	Standard multinomial

Classification

Practical implementation

Tree regression

Tree regression in “default” mode

```
* Load intial dataset
sysuse boston, clear

* Form the train and test datasets
get_train_test , dataname("boston") split(0.80 0.20) split_var(svar) rseed(101)

* Form the target and the features
global y "medv"
global X "zn indus chas nox rm age dis rad tax ptratio black lstat"

* Run tree regression in default mode
use boston_train, clear
r_ml_stata_cv $y $X  , ///
mlmodel("tree") data_test("boston_test") ///
default prediction("pred") seed(10)
```

Results

```
------------------------------------------------------------------------------------------
Learner: Tree regression

Dataset information

Target variable = "medv"                    Number of features   = 12
N. of training units = 405                  N. of testing units = 101
N. of used training units = 405             N. of used testing units = 101
------------------------------------------------------------------------------------------
Parameters

Tree depth = largest tree possible
------------------------------------------------------------------------------------------
Validation results

MSE = mean squared error                    MAPE = mean absolute percentage error
Training MSE = 0                            Testing MSE = 49.644951
Training MAPE % = 0                         Testing MAPE % = 21.923632

------------------------------------------------------------------------------------------
```

Tree regression in “non-default” mode

```
* Run tree regression with specific tree depth
cap rm CV.dta
use boston_train, clear
r_ml_stata_cv $y $X , ///
mlmodel("tree") data_test("boston_test") ///
prediction("pred") tree_depth(3) cross_validation("CV") ///
n_folds(5) seed(10)
```

Results

```
--------------------------------------------------------------------------------
Learner: Tree regression

Dataset information

Target variable = "medv"                    Number of features  = 12
N. of training units = 405                  N. of testing units = 101
N. of used training units = 405             N. of used testing units = 101
--------------------------------------------------------------------------------
Cross-validation results

Accuracy measure = explained variance       Number of folds = 5
Best grid index = 0                         Optimal tree depth = 3
Training accuracy = .84580618               Testing accuracy = .2984019
Std. err. test accuracy = .65469445
--------------------------------------------------------------------------------
Validation results

MSE = mean squared error                    MAPE = mean absolute percentage error
Training MSE = 14.502344                    Testing MSE = 40.720762
Training MAPE % = 16.141842                 Testing MAPE % = 21.355281

--------------------------------------------------------------------------------
```

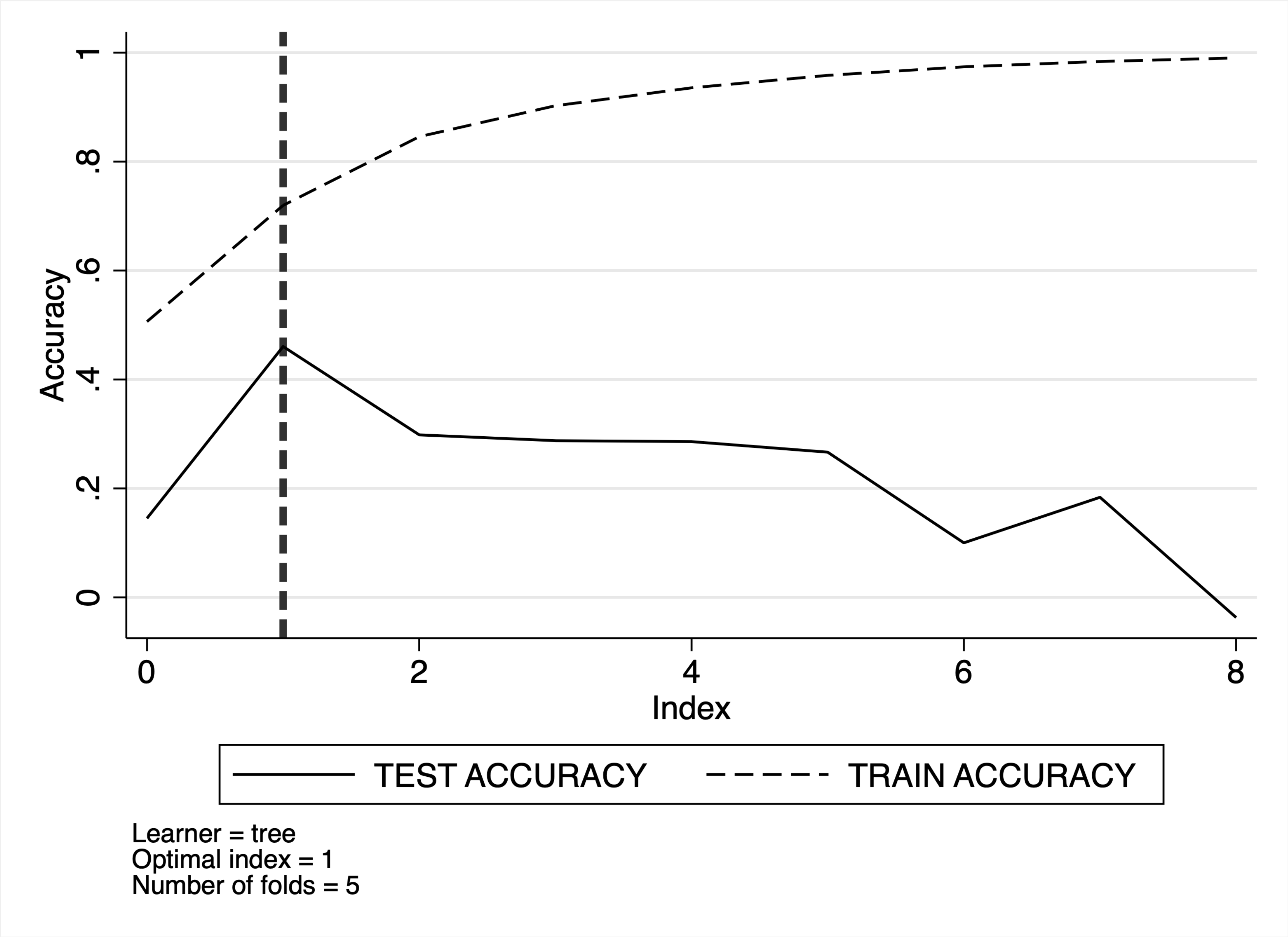
Tree regression in “cross-validation” mode

```
* Run tree regression with cross-validated tree depth
cap rm CV.dta
use boston_train, clear
r_ml_stata_cv $y $X , ///
mlmodel("tree") data_test("boston_test") ///
prediction("pred") tree_depth(1 2 3 4 5 6 7 8 9) cross_validation("CV") ///
n_folds(5) seed(10) graph_cv
```

Results

```
--------------------------------------------------------------------------------
Learner: Tree regression

Dataset information

Target variable = "medv"                    Number of features  = 12
N. of training units = 405                  N. of testing units = 101
N. of used training units = 405             N. of used testing units = 101
--------------------------------------------------------------------------------
Cross-validation results

Accuracy measure = explained variance       Number of folds = 5
Best grid index = 1                         Optimal tree depth = 2
Training accuracy = .71966095               Testing accuracy = .4605888
Std. err. test accuracy = .35495267
--------------------------------------------------------------------------------
Validation results

MSE = mean squared error                    MAPE = mean absolute percentage error
Training MSE = 24.584194                    Testing MSE = 31.301179
Training MAPE % = 19.328019                 Testing MAPE % = 20.389068

--------------------------------------------------------------------------------
```

Graph of cross-validation results



Out-of-sample prediction

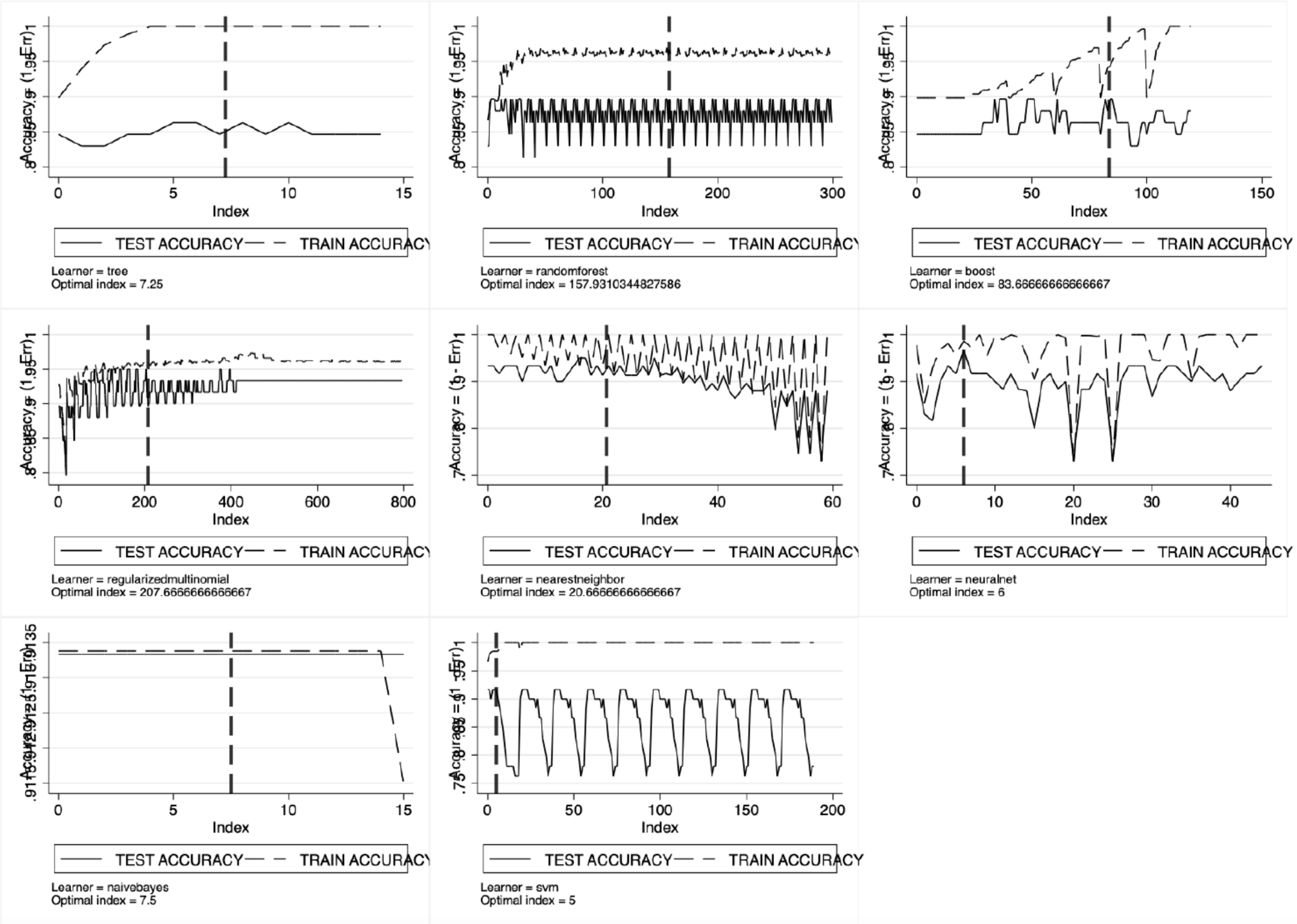
```
. gen id=_n
. line medv pred id if svar==2
```

Example

Comparing multiple learners

Guessing whether a “new” car is a “foreign” or “domestic” one based on a series of characteristics, including price, number of repairs, weight, etc

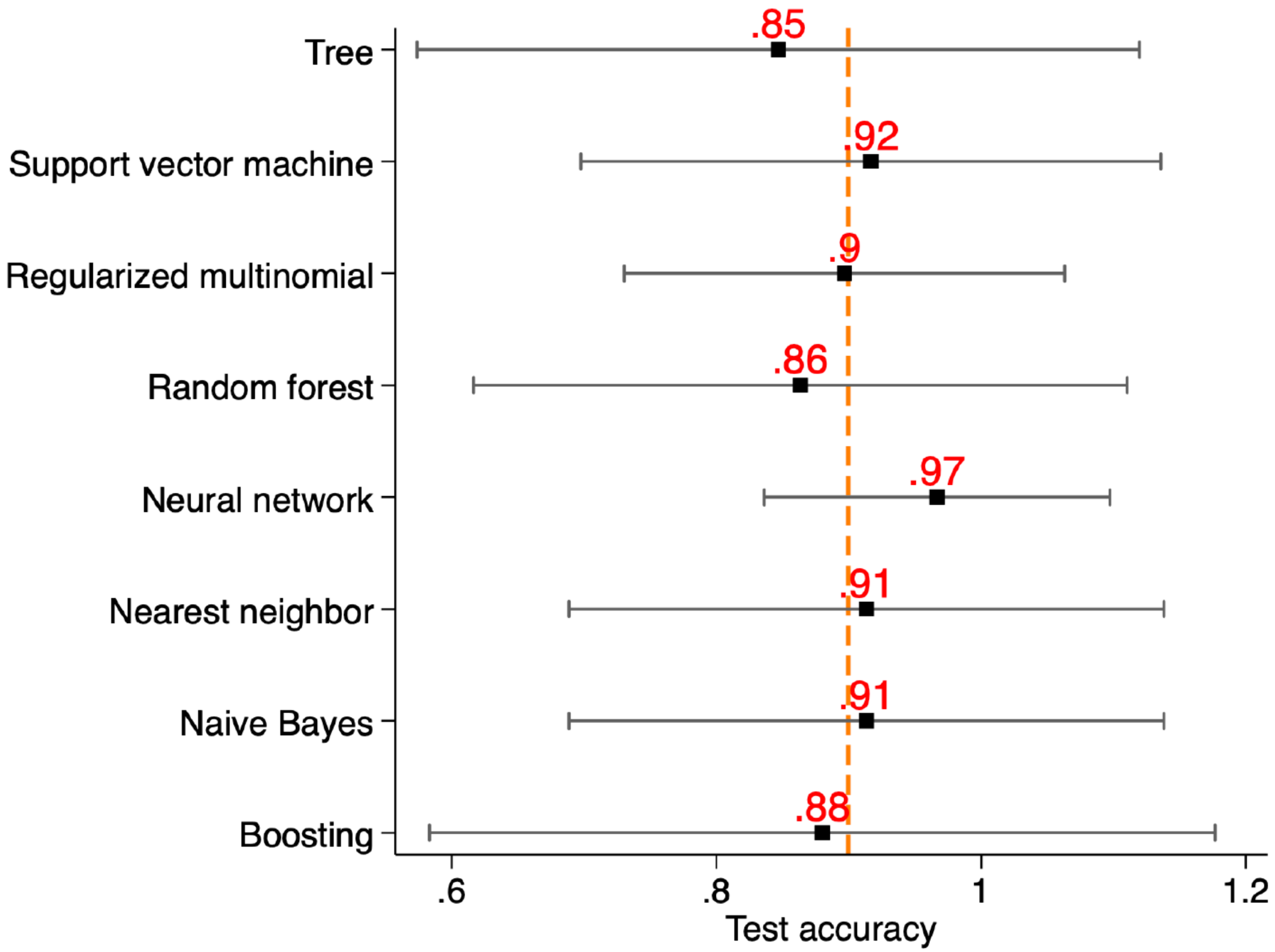
Cross-validation



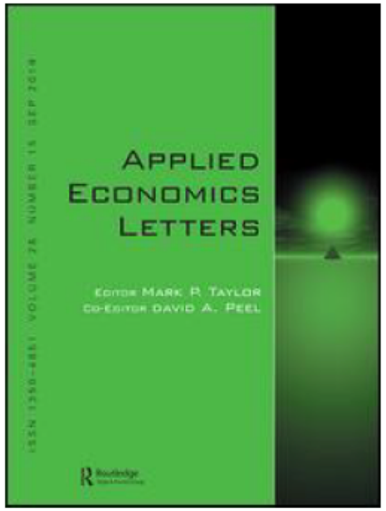
Cross-validation maximum of the classification test accuracy over a grid of learners' tuning parameters.

Accuracy measure: “error rate”

Comparing learner performance



Forest plot for comparing mean and standard deviation of different learners. Classification setting



Routledge
Taylor & Francis Group

Applied Economics Letters

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/rael20

Improving econometric prediction by machine learning

Giovanni Cerulli

References

- Cerulli, G. 2020. *C_ML_STATA: Stata module to implement machine learning classification in Stata*. Statistical Software Components, Boston College Department of Economics. Available at: https://econpapers.repec.org/software/bocbocode/s458830.htm
- Cerulli, G. 2020. *R_ML_STATA: Stata module to implement machine learning regression in Stata*. Statistical Software Components, Boston College Department of Economics. Available at: https://econpapers.repec.org/software/bocbocode/s458831.htm
- Cerulli, G. 2020. *A super-learning machine for predicting economic outcomes*, MPRA Paper 99111, University Library of Munich, Germany, 2020
- Cerulli, G. 2020. Improving econometric prediction by machine learning, *Applied Economics Letters*, Forthcoming.
- Gareth, J., Witten, D., Hastie, D.T., Tibshirani, R. 2013. *An Introduction to Statistical Learning : with Application in R*. New York, Springer
- Raschka, S., Mirjalili, V. 2019. *Python Machine Learning*. 3rd Edition, Packt Publishing.