Learnipy: a Repository for Teaching Machine Learning Without Coding

Fabio Celli
Maggioli s.p.a.
Santarcangelo di Romagna, Italy
Email: fabio.celli@maggioli.it

Cristiano Casadei
Maggioli s.p.a.
Santarcangelo di Romagna, Italy
Email: cristiano.casadei@maggioli.it

***Abstract*—Machine learning is a central expertise in our societies, but it is often not taught in study courses that are not directly associated with technical education. To address this problem, we have designed a system, called Learnipy, to make machine learning accessible and easy to use, independently of their background. Learnipy is an open source Python library designed to make machine learning accessible and easy to use for high school students. It is self-contained, portable, and based on popular Python libraries such as Pandas, Scipy, Numpy, Scikit-Learn, and TensorFlow. Learnipy can take as input tabular data in CSV format or images in ZIP format. It is designed to automatically analyze a dataset and set the default values accordingly, reducing the concepts needed to run algorithms to the core. Learnipy has been used in many courses, from bootcamps to online teaching, and students of very different ages and backgrounds have become able to understand the basic concepts of machine learning and call functions to analyze data without properly coding. We are open to collaborations on Learnipy, which is an open source project.**

Keywords*—*Machine Learning, Deep Learning, Teaching

I. Motivation and Background

The emergence of corporate academies [9] yields companies to share their technical expertise with educational institutions at various levels, from high school to universities. Machine learning is a central expertise in our societies also for study courses that are not directly associated to technical education. To overcome this problem we designed a system to make machine learning accessible and easy to use for high school students, independently on their background. We developed a system that is self-contained (one file), portable and 100% written in Python. It is based on Pandas [7] for data management, Scipy and Numpy for data analysis [3], Scikit-Learn for machine learning [6] and Tensorflow for deep learning [10]. The code is open source, tested on Google Colab, released under MIT license (Commercial use, Modification, Distribution, Private use are permitted, Liability is yours, No software warranty) and published in a Github repository[1].

II. Documentation

In this section we provide a short description of the types of data that Learnipy can take as input and the main functions that we included in the program to process the data, divided by type.

A. *Data formatting*

Learnipy can take as input 2 types of data: tabular data in .csv and images. Tables can contain text inside cells and the csv must be a comma-separated file. In the table the target column can be named 'class' or defined with -d.c= option. The text column can be named 'text' in the .csv file or defined with -d.s= option. Images must be compressed in a .zip file without encryption that must contain .png or .jpg files. the files names must be comma-separated and contain an ID and class, in a specific position and separated by commas. Example: imgID,class,.jpg.

Learnipy is designed to automatically analyze a dataset and set the default values accordingly to simplify the execution, reducing the concepts needed to run algorithms to the core. Here we report the list of functions and algorithms that are provided in Learnipy.

B. *data management*

A set of predefined functions that allow the management of data.
-d.t=c define type of task. c=classification, r=regression
-d.x=n,m,o define the columns to exclude. n,m,o=names of columns to exclude
-d.k=n,m,o define the columns to keep. n,m,o=names of columns to keep
-d.s=n define the string column treated as text. n=name of text column
-d.c=n define the column of the target class. n=name (for .csv) or index (for .zip) of class column
-d.r=0 do not use feature reduction, keep original features (not applicable with -d.save)
-d.f=c_v filter. keep only rows of column c with value v
-d.b=0.5 resample rows. if value is lower than 1, it subsamples a percentage of rows without duplicates. if greater than 1it enables bootstrapping with duplication
-d.m=1 fill class missing values. 1=replace all missing values in class with mean/mode (otherwise are deleted by default)
-d.g=c_a—s group rows by column c (must be nominal). keeps only numeric columns aggregated as a=average or s=sum

[1]https://github.com/facells/learnipy

-d.viz print pca-projected 2d data scatterplot and other visualizations
-d.md model details. prints info on algorithm parameters and data modeling
-d.fdst print info on feature distribution
-d.data show preview of processed data
-d.save save model as .h4 (machine learning) or .h5 (deep learning) file
-d.pred use model to make predictions on new data
-d.export=f export processed data in csv. f=filename.csv

C. data generation

A set of functions to generate data.
-g.d=132 generate an artificial tabular dataset, create gen.csv. 1=num instances x1000, 3=num features x10, 2=num informative features x10

D. preprocessing

A set of functions to preprocess data, especially text data [12]:
-p.ir instance position randomization, applies to the training set
-p.cn class normalize. turn numeric class to range 0-1
-p.fn feature normalize, turn features to range 0-1 (applied by default with some nn, sgd and nb)
-p.tl text to lowercase
-p.tc text cleaning. removes non alphanum char and multiple spaces
-p.trs text regex stopwords. removes words from length 1 to length 3
-p.tsw=a,b text stopwords. removes stopwords, a,b=stopwords list, no spaces allowed.

E. feature reduction

A set of functions for the compression of data [1].
-r.svd=5 singular value decomposition. turn sparse label matrix to dense and sync. 5=number of features
-r.lsa=5 latent semantic analysis. turn sparse word/char matrix to dense and sync. 5=number of features

F. feature extraction

A set of functions to extract features from unstructured data, including text [11] and images [8].
-x.ng=23cf4 ngrams. turn text ngrams matrix and apply lsa. 2=min, 3=max, c=chars—w=words, f=freq—t=tfidf, 4=num x 100
-x.tm=5 text token matrix. turn text into word frequency matrix. 5=number of features
-x.ts=5 text token sequences. columns are padded sequences of words. 5=number of features
-x.cm=5 text char matrix. turn text into character frequency matrix. 5=number of features
-x.bert text embeddings. 768 features from text to a dense matrix with multi-language bert transformer model
-x.mobert text embeddings. 512 features from text to a dense matrix with multi-language mobile bert transformer model
-x.d=e text extraction from custom dictionary. e=dictionary. check https://github.com/facells/learnipy/tree/main/resources
-x.rsz[=32] image resize custom feature extraction. 32=size 32x32, default 16x16 (768 features)
-x.resnet image extraction. 2048 features from pre-trained imagenet model
-x.vgg image extraction. 512 sparse features from pre-trained imagenet model
-x.effnet image extraction. 1408 dense features from pre-trained imagenet model

G. unsupervised learning

A set of functions of techniques of unsupervised learning, including correlation analysis, clustering, association rule learning [5].
-u.km=2 kmeans, centroid clustering. add a new colum to dataset. results in log.txt. 2=num clusters
-u.optics optics, density clustering. add a new colum to dataset. results in log.txt
-u.msh mshift, density clustering. add a new colum to dataset. results in log.txt
-u.ap affinity propagation exemplar clustering. add a new colum to dataset. results in log.txt
-u.som self organising map, neural network clustering. add a new colum to dataset. results in log.txt
-u.arl association rule learning with apriori. prints results in log.txt
-u.corr=s correlation rankings and p-values. s=spearman (monotone+linear), p=pearson (linear). prints results in log.txt
-u.corm=s correlation matrix. s=spearman (monotone+linear), p=pearson (linear). prints results in log.txt

H. outlier detection

A set of functions that performs anomaly detection and removes rows with outlier values [2].
-o.if isolation forest. find and remove outliers using random forest regions
-o.mcd minimum covariance determinant with ellipsis envelope. find and remove outliers using gaussian distribution
-o.lof local outlier factor. find and remove outliers using optics less dense regions

I. supervised learning

A set of classification functions including machine learning and deep learning [4]:
-s.base majority baseline for classification and regression
-s.nb probabilistic models. complement naive bayes for classification, bayes ridge for regression
-s.lr linear regression and logistic regression
-s.lcm linear combination models, linear discriminant classifiction and partial least squares regression
-s.sgd linear modeling with stochastic gradient descent
-s.knn k nearest neighbors classification and regression
-s.dt decision trees and regression trees
-s.mlp multi layer perceptron
-s.svm[=p3] svm (rbf kernel by default). p=polynomial kernel—r=rbf kernel (default), 3=kernel degrees
-s.rf ensemble learning, random forest
-s.ada ensemble learning, adaboost based on samme.r algorithm
-s.xgb ensemble learning, xgboost
-s.nn=f[51] deep learning. f=feedfwd, i=imbalance, r=rnn, l=lstm, b=bilstm, g=gru, c=cnn. 5= x10 units, 1=num layers

J. time series forecasting

A set of functions for statistical time series forecasting [13].
-t.arma auto regression moving average
-t.arima auto regression integrated moving average
-t.sarima seasonal auto regression integrated moving average
-t.hwes Holt-Winters exponential smoothing

K. Evaluation

Options to define evaluation function:
-e.tts=0.2 train-test split. 0.2=20

L. Usage

Learnipy has been designed to run on Google Colab, an example of instructions to run the program is as follows:

```
%run learnipy.py '-d.t=c -x.tm=700
-d.save -s.nn=f' traindata.csv
```

This example runs Learnipy to extract features from text contained in a tabular training data (traindata.csv), generating 700 columns of a token matrix (-x.tm=700) and perform a classification (-d.t=c) with a feedforward neural network (-s.nn=f) that is constructed automatically with default options, then saving the model (-d.save), tht will produce a file of the type model-options.h5.
Then it is possible to make predictions on new data with the following line of code:

```
%run learnipy.py '-d.pred'
model-options.h5 testdata.csv
```

The name of the model will contain the options used for training. Models can have .h5 (deep learning) or .h4 (machine learning) extension. With just one option (-d.pred) it is possible to generate predictions on new data. It is possible to use the program on the following notebook on Google Colab[2].

III. Conclusion

We developed Learnipy, an open source library in Python designed to make machine learning accessible and easy to use for high school students, independently on their background. We used this software in many courses, from bootcamps to online teaching, and students of very different ages and backgrouds become able to understand the basic concepts of machine learning and call functions to analyze data without properly coding.

Being an open source project, we are open to collaborations.

References

[1] Shaeela Ayesha, Muhammad Kashif Hanif, and Ramzan Talib. Overview and comparative study of dimensionality reduction techniques for high dimensional data. *Information Fusion*, 59:44–58, 2020.

[2] Azzedine Boukerche, Lining Zheng, and Omar Alfandi. Outlier detection: Methods, models, and classification. *ACM Computing Surveys (CSUR)*, 53(3):1–37, 2020.

[3] Eli Bressert. Scipy and numpy: an overview for developers. 2012.

[4] Danilo Bzdok, Martin Krzywinski, and Naomi Altman. Machine learning: supervised methods. *Nature methods*, 15(1):5, 2018.

[5] Florian Hahne, Wolfgang Huber, Robert Gentleman, Seth Falcon, R Gentleman, and VJ Carey. Unsupervised machine learning. *Bioconductor case studies*, pages 137–157, 2008.

[6] Oliver Kramer and Oliver Kramer. Scikit-learn. *Machine learning for evolution strategies*, pages 45–53, 2016.

[7] Wes McKinney et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9, 2011.

[8] Dong Ping Tian et al. A review on image feature extraction and representation techniques. *International Journal of Multimedia and Ubiquitous Engineering*, 8(4):385–396, 2013.

[9] Donatella Pinto. Corporate academy: il nuovo perimetro. *Corporate Academy: il nuovo perimetro*, pages 14–19, 2022.

[10] Sebastian Raschka and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.

[11] Foram P Shah and Vibha Patel. A review on feature selection and feature extraction for text classification. In *2016 international conference on wireless communications, signal processing and networking (WiSPNET)*, pages 2264–2268. IEEE, 2016.

[12] S Vijayarani, Ms J Ilamathi, Ms Nithya, et al. Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, 5(1):7–16, 2015.

[13] Richard Webby and Marcus O'Connor. Judgemental and statistical time series forecasting: a review of the literature. *International Journal of forecasting*, 12(1):91–118, 1996.

[2]try it on https://colab.research.google.com/drive/1DfDp2VFaTTMz_B6uLrOdWKQkrer32S9M?usp=sharing