An overview and comparison of free Python libraries for data mining and big data analysis

I. Stančin* and A. Jović *

* University of Zagreb Faculty of Electrical Engineering and Computing / Department of Electronics, Microelectronics, Computer and Intelligent Systems, Unska 3, 10 000 Zagreb, Croatia
stancin.igor@gmail.com, alan.jovic@fer.hr

***Abstract* - The popularity of Python is growing, especially in the field of data science. Consequently, there is an increasing number of free libraries available for usage. The aim of this review paper is to describe and compare the characteristics of different data mining and big data analysis libraries in Python. There is currently no paper dealing with the subject and describing pros and cons of all these libraries. Here we consider more than 20 libraries and separate them into six groups: core libraries, data preparation, data visualization, machine learning, deep learning and big data. Beside functionalities of a certain library, important factors for comparison are the number of contributors developing and maintaining the library and the size of the community. Bigger communities mean larger chances for easily finding solution to a certain problem. We currently recommend: pandas for data preparation; Matplotlib, seaborn or Plotly for data visualization; scikit-learn for machine leraning; TensorFlow, Keras and PyTorch for deep learning; and Hadoop Streaming and PySpark for big data.**

Keywords - data science, python, data mining, machine learning library, big data analysis, framework

I. INTRODUCTION

Data mining (DM) deals with preparation of data obtained from various information sources (e.g. databases, text files, streams) as well as data modeling using a variety of techniques, depending on the goal that one wants to achieve (e.g. classification, clustering, regression, association rule mining, etc.). DM uses machine learning (ML) techniques to discover new knowledge from the existing information. DM is, nowadays, mostly considered within the wider scope of data science, which also encompasses statistics, big data techniques and data visualization. Data preparation is a vital step in the process of data analysis, and it includes data preprocessing and data manipulation (sometimes also called wrangling). Preprocessing aims at cleaning, integrating, transforming and reducing the original raw data so that it can become usable for data analysis, while wrangling transforms the preprocessed dataset into a data format that can be easily manipulated by the data modeling algorithms.

The use of Python in the area of data science has reached unprecedented levels, especially in the area of freely available tools and libraries. In a poll published in May 2018 by the authoritative portal KDNuggets [1], under the category "Top Analytics, Data Science, Machine Learning Tools", it was found that Python is used by 65.2% of roughly 2000 participants, compared to 52.7% for RapidMiner and 48.5% for R, its two major competitors. In practical perspective, in the last three years, Python has become the programming language of choice for the data science community, with R being the second choice. The Python's popularity probably stems from its relative ease of use (even for non-computer scientists), huge ecosystem consisting of a number of libraries for every aspect of data science and its reliance via *NumPy* and *SciPy* wrappers on the fast implementations of a large number of scientific algorithms written in C and Fortran.

In our previous work from 2014, we have provided a comparison of freely available tools for general DM [2]. At the time, Python based tools were still not mature enough, while R, RapidMiner, Weka and Knime were at the forefront of the most popular tools. In contrast, the aim of this work is to provide an overview and comparison of various existing Python based libraries for data science. Specifically, we focus on six groups of libraries: Python core, data preparation, data visualization, machine learning, deep learning and big data. We estimate the libraries' significance based on a detailed analysis of their capabilities, the number of contributors and the community size. Since deep learning is a rather recent development in data science, but already with a steady and growing tools' support in Python, we include these libraries, too.

II. AN OVERVIEW AND COMARISON OF LIBRARIES

A. Core libraries

Many DM and ML tasks in Python are based on fast and efficient numerical and vectorized computing with *NumPy* [3] and *SciPy* [4] libraries. Many functionalities from these libraries are actually wrappers around the *Netlib* [5], secure and robust scientific implementations of algorithms. Main advantage of *NumPy* and *SciPy* is their ability of performing efficient vectorized computing and broadcasting over *n*-dimensional arrays.

The other advantage of using Python in this field is the fact that it is relatively easy to connect third party code into the Python interpreter. Probably the most commonly used library for that purpose in the of DM is *Cython* [6]. *Cython* is a language built on top of Python that also supports calling C functions and having C type of variables and classes. The usage of *Cython* can make some critical parts of code several times faster.

All three aforementioned libraries have a stable code and are in constant maintaining and development. Table 1 shows useful information about libraries' "reputation" on GitHub, a web-based hosting service for version control [7], using the number of stars, forks, contributors and activity on the library repository. Activity is shown through the number of contributing authors and the number of commits in the last month.

B. Data preparation

Since everything in the field of data science is based on data, there is a need for data preparation libraries. Currently the best and most used Python library in this field is *pandas* [8]. *pandas* has a wide range of capabilities for input/output data formats, like Excel, csv, Python/*NumPy*, HTML, SQL and more. Furthermore, *pandas* has powerful querying possibilities, statistic calculations and basic visualizations. It has a rich documentation, but a bit confusing syntax, which is often pointed out as its most significant flaw.

Every other library in this field has much bigger issues than *pandas*. *PyTables* [9] and *h5py* [10] accept only HDF5 data type, which is a huge limitation for general usage. There are several more similar libraries (e.g. *Tabel* [11]), but none of them can be competitive to *pandas*, for now.

C. Data visualization

Table 3 shows a comparison of data visualization libraries. *Plotly* [12] has support for most of the standard plots that are used in DM and ML. *seaborn* [13] has a few capabilities less than *Plotly*, and *Matplotlib* [14] has a few less than *seaborn*. Although there are differences between these three libraries, they all have the main plotting capabilities. *Bokeh* [15] and *ggplot* [16] have the fewest options and are the least used libraries.

Matplotlib is a Python implementation of the MATLAB-like plots and is written on a low level, with a lot of possibilities for customization. Its syntax can be a bit confusing at first, but once one masters its main concepts, it is easy to draw pretty much any graph. *seaborn* is built on top of *Matplotlib* and is easier for usage and learning for beginners than *Matplotlib*. Although it is easier to use, in the cases of some complex graphs with a need for a lot of customization, it is possible that *seaborn* would be an infeasible option.

Plotly seems to be the most powerful library in data visualization field. Its main flaw is a relatively unintuitive syntax, making it harder to learn for beginners. However, the flaw is compensated with a very rich documentation providing a lot of examples. It is possible to integrate *Plotly* graphs into webpages with *Dash* [17]. *Bokeh* is intended for integration of interactive plots into webpages, where a user can explore data himself. g*gplot* is the Python's implementation of R's way of plotting. It has a limited documentation and sacrifices customization in order to have a simple and straightforward code.

Although all of the libraries in this group are relatively popular based on the data presented in Table 1, we must mention that *ggplot* has not been maintained or developed in the last two years.

Table 1. Information about libraries from GitHub

Library	Stars	Forked	Contributors	Activity
NumPy	9621	3318	726	28 (103)
SciPy	5418	2690	685	21 (101)
Cython	3833	799	275	10 (85)
pandas	18134	7233	1407	65 (217)
PyTables	801	164	60	0 (0)
h5py	1042	288	98	3 (6)
Tabel	11	0	1	1 (1)
Matplotlib	8688	3966	787	20 (218)
seaborn	5722	905	87	0 (0)
Plotly	4569	1068	68	5 (38)
Bokeh	8969	2398	346	11 (52)
ggplot	3429	539	13	0 (0)
scikit-learn	33337	16358	1253	38 (94)
mlpy	5	2	1	0 (0)
Shogun	2312	891	153	8 (57)
mlxtend	2033	475	46	3 (17)
TensorFlow	120547	72008	1834	194 (1888)
Keras	38196	14584	773	20 (53)
PyTorch	24781	5878	934	152 (913)
Caffe	27016	16335	267	0 (0)
Caffe2	8407	2130	196	0 (0)
mrjob	2367	570	82	3 (143)
Dumbo	1037	161	6	0 (0)
Hadoopy	245	62	3	0 (0)
Pydoop	168	53	11	1 (18)
Spark (PySpark)	20576	18057	1330	78 (246)
Hadoop (Streaming)	8567	5360	155	58 (456)

Note: 1) activity represents: number of contributing authors (number of commits) in the last month; 2) data is from February 14th, 2019

D. Machine Learning

scikit-learn [18] is the most popular Python library for machine learning. Beside it, there are also *mlxtend* [19], a new and small library that includes only a few basic algorithms, and *Shogun* [20], which is primarily written in C++, but there is an available Python wrapper for all of its functionalities. *Shogun* has more algorithms than *mlxtend*, but far less than *scikit-learn*. There are only a handful of algorithms that *Shogun* has implemented and *scikit-learn* does not, which can be seen in Table 2. There is also a library called *mlpy* [21], which is not listed in Table 2. The reason for its absence is that it is a small library, similarly to *mlxtend*, but it does not have any well known algorithm implemented that other libraries do not have.

scikit-learn has an advantage in the number of algorithms implemented in most categories in Table 2. *Shogun's* advantage over the other libraries is in the number of algorithms that implement different kinds of trees. Although *mlxtend* is a small library, it is the only library with implemented association rule algorithms and stacking ensemble learning. The lack of these algorithms can be considered a huge omission by *scikit-learn* and *Shogun*. The same goes for inductive rule learners, full Bayesian network, rotational forest and fuzzy c-means clustering, which are not implemented in any of the listed libraries.

Table 2. Comparison of machine learning libraries

Category	Supported algorithms	scikit-learn	mlxtend	Shogun
Feature selection	Filters	+ (many methods)	+ (one method)	-
	Wrappers	+ (many methods)	+ (two methods)	-
Feature transformation	Discretization	+	-	-
	Normalization	+	-	+
	PCA	+ (several methods)	+	+
	ICA	+ (several methods)	+	+ (various)
	MDS	+	-	+
	Manifold learning	+	-	+
	SVD	+	-	-
	Random projections	+	-	-
	LDA (for dimensionality reduction)	+	+	+
	GDA (Kernel Fisher Discriminant Analysis, for dimensionality reduction	+	-	-
	Factor analysis	+	-	+
	tSNE	+	-	-
	others	+	-	+
Decision tree learner	ID3	-	-	+
	C4.5	-	-	+
	CART	+ (optimized)	-	+
	CHAID	-	-	+
	RelaxedTree	-	-	+
	others	-	-	+ (Conditional probability tree, Nobody tree)
Bayesian classifiers	Naïve Bayes	+ (various distribution assumtions)	-	+
	others	+ (ComplementNB)	-	-
Function based classification	LDA classifier	+	-	-
	Logistic regression	+	+	-
	GDA classifier	+	-	-
	Elastic net	+	-	-
	Others	+	+ (Softmax regression)	-
Instance based learning	kNN	+ (several)	-	+ (several)
	Nearest centroid classifier	+	-	+
Regression analysis	Ordinary least squares linear regression	+	+	-
	Ridge regression	+	-	-
	Kernel ridge regression	+	-	+
	PLS regression	+	-	-
	Lasso (and variations)	+	-	-
	Least angle regression	+	-	-
	Polynomial regression	+	-	-
	others	+ (Bayesian regr. Robustness regr., Isotonic regr. …)	-	-
ANN	Perceptron	+	+	+
	MLP classification and/or regression	+	-	+
	Restricted Bolzman Machine	+	-	-
	Building your own NN	-	-	+
	others	-	+ (Adaline, Multilayer perceptron)	+ (Averaged perceptron)
SVM	SVC (or NuSVC)	+	-	+
	SVR (or NuSVR)	+	-	+
	OneClassSVM	+	-	+
	LaRankSVM	-	-	+
	others	+ (RBF kernel SVM)	-	+ (NewtonSVM, SVMSGD, LPBoost, MKLRegression)
Ensemble learning	Bagging	+	-	+
	AdaBoost	+	-	-
	Random forest	+	-	+
	Extremely randomized trees	+	-	-
	Totally randomized trees	+	-	-
	Gradient boosting	+	-	+
	stacking	-	+	-
	Majority voting	+	+	+
	others	+ (IsolationForest, weighted average voting)	+ (stacking regressor,	+ (MeanRule, Combination Rule)

Hierarchical clustering	AgglomerativeClustering	+ (Ward; single, average and complete linkage strategies)	-	+ (single linkage)
	BIRCH	+	-	-
Centroid (partition) clustering	k-means	+	+	-
	Mean Shift	+	-	-
Distribution based clustering	EM clustering	+ (Gaussian mixture model)	-	+ (Gaussian micture model)
	Affinity propagation	+	-	-
	Spectral clustering	+	-	-
Density based clustering	DBSCAN	+	-	-
Association rules (unsupervised)	Apriori	-	+	-
	Association rules	-	+	-
Evaluation methods and metrics	Holdout	+	+	-
	Cross-validation	+	-	+
	Regression evaluation: MSE, MAE, Pearson's correlation coefficient	+	-	-
	Classification evaluation: TP, FP, FN, TN, confusion matrix, accuracy, precision, recall, F1…	+	+	+
	Clustering evaluation: Adjusted rand index, Normalized mutual information, Silhouette Coefficient, Calinski-Harabasz index…	+	-	+ (Normalizer mutual information)
	ROC, PRC, Lift chart, Cost-benefit	+ (ROC, PRC), liftchart and cost-benefit in scikit-plot	-	+ (ROC, PRC)
	other	+ (OVO, OVR, GridSearchCV, RepeatedKFold, …)	+ (bootstrap, lift score, …)	+ (ECOCStrategy , OVO, OVR, GridSearch, …)

From popularity shown in Table 1, we can see that *scikit-learn* has a huge community, while *mlpy* has a very small community. It should also be mentioned that *scikit-learn* has the best documentation, which is intuitive for usage.

E. Deep learning

Table 4 shows functionalities that certain deep learning library have implemented. Basic functionalities are implemented in all four available libraries. *Caffe* [22] does not have much more than these basic functionalities and its documentation is not intuitively structured. *Caffe* has its new version – *Caffe2* [23], but it has a similar number of functionalities. *TensorFlow* [24] (*TF*) is developed by Google Brain, it has a good documentation, a lot of functionalities beside the basics and it is possible to make code very customable. Since it is written as a low level library, it is bit harder to master. TensorBoard is a visualization tool that comes with all the standard installations of TF. It allows users to monitor their models, parameters, losses, and much more.

Keras [25] is built on top of *TF*. Coding in *Keras* is therefore on a higher level. The cost for that is a harder customization of code. It is well known that customization and tweaking of code is much easier when coding at a low level. *PyTorch* [26] (*PT*) is developed and used by Facebook. It was developed more recent than *TF*, but its community is growing fast. *PT* is dynamic and it runs code in a more procedural fashion, while in *TF*, one first needs to design the whole model and then run it within a *Session*. Because of this, it is much easier to debug code in *PT*. *PT* has more "pythonic" codes, it is easier to learn and easier to use for quick prototyping. *PT* and *Keras* also have good documentations.

Table 3. Comparison of data visualization libraries

Plot type	Matplotlib	seaborn	Plotly	Bokeh	ggplot
Line chart	+	+	+	+	+
Histograms	+	+	+	-	+
Bar	+	+	+	+	+
Scatterplots	+	+	+	+	+
Boxplot	+	+	+	-	-
Contures	+	+	+	-	-
Filled polygons	+	-	+	+	-
Spectrogram	+	-	+	-	-
Violin plot	+	+	+	-	-
Pairplot	-	+	-	-	-
Heatmap	-	+	+	+	-
Matrix clustermap (dendogram)	-	+	+	-	-
Regression plot	-	+	-	-	+
Joint plot	-	+	+	-	-
Polar plot	+	-	+	-	-
3D	+	-	+	-	-
Interactive graphs and animations	+	-	+	+	-
Others	+	+	+	+	-

NOTE: If there is minus in some column, it does not necessarily mean that it is not possible to do it, but that there is no direct function for a wanted plot (for example, pairplot is possible to create with Matplotlib with several lines of code and a scatterplot)

Table 4. Comparison of deep learning libraries

Category	Supported method	TensorFlow	Kearas	PyTorch	Caffe
Layers	Conv1D, Conv2D, Conv3D	+	+	+	+
	ConvTranspose1D, ConvTranspose2D, ConvTranspose3D,	+	+	+	+
	SeparableConv1D, SeparableConv2D	+	+	-	-
	MaxPool1D, MaxPool2D, MaxPool3D	+	+	+	+
	AvgPool1D, AvgPool2D, AvgPool3D	+	+	+	+
	AdaptivePool (all combinations)	-	-	+	-
	GlobalPool	-	+	-	-
	Dense	+	+	+	+
	Dropout	+	+	+	+
	Flatten	+	+	-	+
	Padding	+	+	+	+
	RNN	+	+	+	+
	LSTM	+	+	+	+
	GRU	+	+	+	-
	Normalization	+	+	+	+
	Noise	-	+	-	-
	others	+	+	+	+
Activation functions	ReLu	+	+	+	+
	ReLu6	+	-	+	-
	PReLu	-	+	+	+
	LeakyReLu	-	+	+	-
	CReLu	+	-	-	-
	ThresholdedReLu	-	+	+	-
	Elu	+	+	+	+
	Selu	+	+	+	-
	Softplus	+	+	+	-
	Softsign	+	+	+	-
	Bias_add	+	-	-	-
	Sigmoid	+	+	+	+
	Hard_sigmoid	-	+	-	-
	Exponential	-	+	-	+
	Linear	-	+	-	-
	Softmax	+	+	+	+
	Tanh	+	+	+	+
	others	+	+	+	+
Losses	MSE	+	+	+	+
	Log_loss	+	-	-	-
	Hinge_loss	+	+	+	+
	Logcosh	-	+	-	-
	Cross_entropy	+	+	+	+
	Poisson	+	+	-	-
	Cosine_distance	+	-	-	-
	Huber	+	-	-	-
	NLLLoss	+	-	+	-
	CTCLoss	+	-	+	-
	KLDivLoss	-	+	+	-
	NCELoss	+	-	-	-
	BCELoss	-	+	+	-
	SoftMarginLoss	-	-	+	-
	CosineEmbeddingLoss	-	-	+	-
	MultiMarginLoss	-	-	+	-
	others	+	+	+	+
Optimizers	GradientDescent (GD)	+	-	-	-
	Proximal GD	+	-	-	-
	StohasticGradientDescent (SGD)	-	+	+	+
	Averaged SGD	-	-	+	-
	RMSprop	+	+	+	+
	Rprop	-	-	+	-
	Adadelta	+	+	+	+
	Adagrad	+	+	+	+
	AdagradDualAveraging	+	-	-	-
	ProximalAdagrad	+	-	-	-
	Adam	+	+	+	+
	AdaMax	-	+	+	-
	Nadam	-	+	-	+
	SparseAdam	-	-	+	-
	L-BFGS	-	-	+	-
	FTRL	+	-	-	-
	Momentum	+	-	-	-
GPU acceleration		+	+	+	+

F. Big data

Currently, the most popular tools for big data are Spark and Hadoop MapReduce. Both are scalable, flexible and fault tolerant tools. They have their own specialized storage system, which allows them to work on clusters of computers. Spark uses the Resilient Distributed Datasets (RDDs), while Hadoop uses the Hadoop distributed file system (HDFS). The main difference between Spark and Hadoop MapReduce is the fact that Spark can work within the RAM memory, while Hadoop always writes on the file system. Hadoop is a good choice in the cases of very large amount of data (larger than the available RAM) and when there is no need for immediate results. In all other cases, Spark is probably a better choice. Although both are written in Java, many big data engineers prefer to use them in combination with Python.

Hadoop Streaming [27] is an interface that allows the usage of any language for MapReduce jobs on Hadoop. The other possibility is to use *mrjob* [28], an open source wrapper around *Hadoop Streaming*. It is actively developed by Yelp and it has a good documentation. A disadvantage of *mrjob* is that it is a simplified framework that does not provide some advanced functionalities and does not have available support for typedbytes, so it is a bit slow in some cases. In contrast, *Dumbo* [29] provides more advanced functionalities. It is also a wrapper around *Hadoop Streaming*, but its documentation is not that rich, which makes it harder to use. It is very similar to *Hadoopy* [30], which also has support for typedbytes serialization of data and is a *Hadoop Streaming* wrapper. *Hadoopy* has a relatively good documentation. *Pydoop* [31] is a wrapper around *Hadoop pipes* (C++ API for Hadoop). There are a few additional Python libraries for Hadoop, but we find that those mentioned above are currently the best options. *Dumbo* and *Hadoopy* have not been maintained or developed for the last 5 years.

Regarding Spark, we are not aware of any other Python library other than *PySpark* [32]. *PySpark* is an API that exposes Spark data processing model to Python.

III. Conclusion

For data preprocessing and manipulation, we recommend the usage of *pandas*. It has a strong community support, a rich offer of functionalities and no serious competition. In the field of data visualization, things are not so clear-cut and the choice of library largely depends on the project. *Plotly* has the most capabilities, *seaborn* is very intuitive and easy to use, while *Matplotlib* offers many possibilities for customization. All three have strong communities.

scikit-learn is the best library in the field of machine learning. It has a very good and intuitive documentation with many examples. It has a large scope of implemented algorithms. Deep learning is a relatively recent field, but with three very good libraries. We recommend the usage of *PyTorch* or *Keras* for quick prototyping and *TensorFlow* for projects which demand a lot of customization.

Hadoop Streaming and *PySpark* are the best libraries to use in the field of big data. Both are APIs for native libraries (Hadoop and Spark), so they have a large community support.

References

[1] KDnuggets, (2019, February 4th), https://www.kdnuggets.com/.

[2] A. Jovic, K. Brkic and N. Bogunovic, "An overview of free software tools for general data mining," 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2014.

[3] T. E. Oliphant, A guide to NumPy, USA: Trelgol Publishing, 2006.

[4] E. Jones, T. E. Oliphant, P. Peterson, et al. SciPy: Open Source Scientific Tools for Python, 2001.

[5] S. Browne, J. Dongarra, E. Grosse and T. Rowan, The Netlib Mathematical Software Repository, D-Lib Magazine, 1995.

[6] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn and K. Smith, Cython: The Best of Both Worlds, Computing in Science and Engineering, 13, 31-39, 2011.

[7] GitHub, (2019, February 14th), https://github.com/.

[8] W. Mckinney, pandas: a Foundational Python Library for Data Analysis and Statistics. Python High Performance Science Computer, 2011.

[9] F. Alted and M. Fernández-Alonso, PyTables: Processing And Analyzing Extremely Large Amounts Of Data In Python, 2003

[10] A. Collette, h5py, (2019, February 4th), https://www.h5py.org/

[11] B. Bergman, Tabel, (2019, February 4th), https://github.com/BastiaanBergman/tabel.

[12] Plotly Technologies Inc., (2019, February 4th), https://plot.ly/.

[13] seaborn, (2019, February 4th), doi: 10.5281/zenodo.883859.

[14] J. D. Hunter, Matplotlib: A 2D graphics environment, Computing In Science & Engineering, 9(3), 90-95, 2007.

[15] Bokeh Development Team, Bokeh: Python library for interactive visualization, 2018.

[16] ggplot, (2019, February 4th), http://ggplot.yhathq.com/.

[17] Dash, (2019, February 4th), https://plot.ly/products/dash/

[18] F. Pedregosa et al., Scikit-learn: Machine Learning in {P}ython, Journal of Machine Learning Research, 12, 2825-2830, 2011.

[19] S. Raschka, MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack, The Journal of Open Source Software, 3(24), 2018.

[20] S. Sonnenburg, et al., The SHOGUN Machine Learning Toolbox, Journal of Machine Learning Research, 11, 1799-1802, 2010.

[21] D. Albanese, R. Visintainer, S. Merler, S. Riccadonna, G. Jurman and C. Furlanello, mlpy: Machine Learning Python, 2012.

[22] Y. Jia, et al. Caffe: Convolutional Architecture for Fast Feature Embedding, MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia, 2014.

[23] Caffe2, (2019, February 4th), https://caffe2.ai/.

[24] M. Abadi, et al., TensorFlow: A system for large-scale machine learning, Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16), 2016.

[25] F. Chollet, Keras, (2019, February 4th), https://github.com/fchollet/keras.

[26] PyTorch, (2019, February 4th), https://pytorch.org/.

[27] Apache, Hadoop Streaming, (2019, February 4th), https://hadoop.apache.org/docs/current/hadoop-streaming/HadoopStreaming.html.

[28] mrjob, (2019, February 4th), https://pythonhosted.org/mrjob/.

[29] Dumbo, (2019, February 4th), https://github.com/klbostee/dumbo

[30] B. White, Hadoopy, (2019, February 4th), https://github.com/bwhite/hadoopy

[31] S. Leo, G. Zanetti, Pydoop: a Python MapReduce and HDFS API for Hadoop., Proceedings Of The 19th ACM International Symposium On High Performance Distributed Computing, pp. 819–825, 2010.

[32] Apache, PySpark, (2019, February 4th), https://spark.apache.org/docs/latest/api/python/index.html