PyDaSci: A Domain Specific Language for Data Analysis

1st Juan Riofrío-Valarezo
School of Mathematical and Computational Sciences
Yachay Tech University
Urcuquí, Ecuador
juan.riofrio@yachaytech.edu.ec

***Abstract*—Data analysis field has grown enormously in the last years with the increase of data obtained and researched methods to analyze it. It can improve the competitiveness or increase the profits of a business, it can show new or better techniques that can be applied in industries or it can improve researches' results. Many analysts know the theory and mathematical aspects of data analysis, but they do not have or have limited programming skills, hindering their job or making them take longer to solve a problem. To address this problem, this work introduces an easy to use domain specific language called PyDaSci for time series forecasting and multivariate data clustering with high level of abstraction. The presented approach offers a way to make models easily and quickly while maintaining efficiency and accuracy. The methods and algorithms supported by this language are mainly SARIMA and Exponential Smoothing in the area of time series forecasting and KMeans, Agglomerative, Spectral and Gaussian Mixture in the field of clustering algorithms.**

***Index Terms*—Data analysis, Domain specific language, Time series, Forecasting, multivariate data, Clustering.**

I. Introduction

Data analysis is one of the most important processes applied in business, industries and researches to obtain more profits, new techniques or better results. Data analysis is a process that refers to a set of methods or procedures applied to collected data to gather new insights that could not be seen with raw data. It is used to improve or help the development of the industry, business or research from where the data was acquired. As there are different types of data, the analysis may not be the same in every case, even if the data is obtained from similar sources the context or what we are looking for may affect the selected method for analysing the data and often involves considerable thought [1]. For this reason there are many methods for addressing data analysis like forecasting, clustering among others and a data analyst should know as many as possible.

An important topic in data analysis comprise time series analysis, contrary to static data, time series are composed of values that change over time. Time series are important for many people due to its presence in many areas as science, engineering, business, finance, economic, health care or even government as mentioned by Liao [2]. Another topic of importance for data analysts is clustering, it is the unsupervised classification of a collection of samples into groups based on their similarities. It groups a given collection of unlabeled data into meaningful groups, labels are associated with clusters, but these labels are data-driven, as they are obtained entirely from the data [3].

In computer science as in data analysis humans need to communicate ideas, either to other humans or to machines. To communicate this ideas we use a language, it can be a General Purpose Language(GPL) as C or Python, or it can be a Domain Specific Language (DSL) like SQL o HTML. DSLs are languages whose instructions are easy and intuitive for a specific domain as defined by Fowler [4]. This programming languages can be used to work in a specific topic in a easier way for people having a certain knowledge of the domain the DSL work on.

This work introduces PyDaSci, a new DSL to work in time series forecasting as well as multivariate data clustering. In this case, the domain the language is going to work on, is data analysis specifically forecasting and clustering. The purpose of the proposed DSL is to make it easier to the user to work in some specific areas of data analysis, the functions and implementations will be intuitive and related to the the theory and terminology used in data science.

II. Related Work

There are several researches in the designing of DSLs for almost any topic, to make it easier for the users if they are not experienced programmers. In this section some of the works that are related to data analysis are presented:

In [5], a DSL used for data analytics is presented. The name of the mentioned DSL is ScalOps, similar to Pig it incorporate the declarative style of SQL and the lowlevel procedural style of MapReduce. On the other hand, it optimizes the runtime for repeated access to data collections like Spark. It is different to other platforms on the following: First, ScalOps provides an explicit loop construct that captures the iteration in the runtime plan. Second, ScalOps uses a recursive query plans to execute iterations in a data-parallel runtime called Hyracks. And third, Hyracks is optimized for computations over in-memory and external memory datasets. This differences allows ScalOps to handle a wider range of analytic over any data.

In [6], the authors showed how a DSL for Educational Data Mining can be developed. The purpose of the DSL is to allow teachers of courses hosted in e-leraning platforms to analyze their own and also students performance by applying data

mining techniques on the data obtained from such platforms. It presents mainly two advantages, first it abstracts low level-details of data analysis techniques and second it is flexible enough to support the elaboration of arbitrary complex new queries.

In [7], a DSL named MMC-BPM is presented. This DSL helps by explicitly modeling associated rules and measurement variables making them more transparent to the final user. It differs from other approaches as domain-related metrics are easier to support due to the data model used to describe data associated with the process elements and the capacity to determine new measurement data. Also the aspect-oriented approach when defining and implementing analysis concerns offers process analysts the favorable circumstances to evolve them regardless of the implementation of the process.

In [8], a clustering DSL is presented, the name of the DSL is DWARF. It mainly provides programming abstraction for data analysts to create new or implement already existing clustering algorithms. It automatically parallels the algorithms,so it is also abstracted from the analyst thereby they do not have to worry of this aspect when programming.

In [9], a new DSL is presented to face the computational power problem when applying machine learning algorithms to large datasets. It is called OptiML and provides a link between heterogeneous parallel hardware and machine learning applications. It implement parallelism abstraction generating efficient parallel code for heterogeneous devices without showing any parallelism or device details to the user.

III. PyDaSci

PyDaSci is a textual DSL designed to address the problem of people working in data analysis being rookie programmers. The language intends to make all the power of Python available to analysts while maintaining the simplicity in its instructions enabling anyone with some expertise in the subject to program in an easier way. PyDaSci will be built on top of Python, an interpreted programming language whose philosophy emphasizes the readability of its code and one of the most used programming languages in Data Science. As an internal DSL, PyDaSci will inherit some properties of the host language (Python). Having said that, the created language will be dynamically typed as Python, meaning that variable type checking is done at runtime as pointed by [10]. Also, as mentioned before, Python is an interpreted programming language; this are other properties inherited by PyDaSci. As an interpreted language, PyDaSci uses pre-compiled routines to quickly and immediately execute each line of code written as defined by [10]. PyDaSci will work with time series represented as a list, and multivariate data represented as a matrix. Some implementations for slicing or extending the time series or the matrices are supported. Forecasting functions will be implemented to make it easier to predict time series future values, plotting will be implemented as well to visualize the results or the time series. Clustering functions will be implemented to group the samples from the multivariate data in different categories. The DSL can be dowloaded from https://github.com/ColdRiver93/PyDaSci.

IV. Results

PyDaSci implement algorithms to analyze both time series and multivariate data but it differentiate both as different data constructs, so each one has their own analysis methods. This methods are explained in more detail below.

A. Time Series Analysis

PyDaSci represent a time serie as a list of elements. This list is similar than in Python, the elements(each observation over time) are separated by comas and enclosed in brackets '[]'. Two time series forecasting methods are supported by PyDaSci, SARIMA and Exponential Smoothing. The functions for creating and fitting the models are presented in 1 followed by the function to forecast new values in 2 where number_f_v means the number of future values to forecast.

$$\begin{aligned} SARIMA(ts\ (p,d,q)\ (p,d,q,s));\\ EXPSM(ts\ trend\ damped\ seasonal\ s\_periods); \end{aligned} \tag{1}$$

$$FORECAST\ model\ number\_f\_v; \tag{2}$$

Regarding time series PyDaSci have also implemented some plotting functions. 3 presents the function to plot time series and 4 presents the function to compare several time series belonging to the same date range.

$$PLOT\ ts\ 'Title'; \tag{3}$$

$$C\_PLOT\ ts_1\ ts_2\ \ldots\ ts_n\ 'Title'; \tag{4}$$

B. multivariate Data Clustering

PyDaSci represent multivariate data as a list of lists or a matrix. This list of lists is similar than in Python, the attributes of each sample are separated by comas and enclosed in brackets '[]' and every sample is separated by comas and again enclosed in brackets'[]' having a list of samples each one with it corresponding attributes. In a matrix perspective each row is a sample and every column is an attribute. Four multivariate clustering methods are supported by PyDaSci, KMeans clustering, Gaussian Mixture clustering, Agglomerative clustering and Spectral clustering. The functions for creating and fitting the different models are presented in 5 where number_clusters is the number of cluster to be divided into, followed by the function to predict the clusters in 6 where obs_matrix is the matrix of observations to be clustered. It is important to mention that Agglomerative and Spectral clustering work in a different way than KMeans and Gaussian Mixture clustering. The first ones use PREDICT2 and cannot predict the cluster of new observations, instead they predict the clusters only of the data that the model was fitted with. While KMeans and Gaussian Mixture use PREDICT, they are fitted with some data and can predict the clusters for the same

data or even predict the clusters of new data presented to the fitted model.

$$\begin{aligned} KMEANS(matrix\ number\_clusters);\\ GAUSSIAN(matrix\ number\_clusters);\\ AGGLOMERATIVE(number\_clusters);\\ SPECTRAL(number\_clusters); \end{aligned} \tag{5}$$

$$PREDICT\ model\ obs\_matrix; PREDICT2\ model\ obs\_matrix; \tag{6}$$

C. Functions for both Time Series and Multivariate Data

There are other implemented functions in PyDaSci that works the same for both data structures. Statistics of the selected data are shown with the STATS command. The function is shown in 7 where 'data' can be a time series or a matrix.

$$STATS\ data; \tag{7}$$

There is also a printing function shown in 8 that can print the time series or matrix passed as argument, it can also print a certain sentence when the text is single quoted.

$$\begin{aligned} PRINT\ data;\\ PRINT\ 'Sentence'; \end{aligned} \tag{8}$$

PyDaSci also let the analysts split and extend time series and matrices. It works similarly to Python where a range of values can be selected from the data where the first number is included and the second number is excluded. The implementation of the splitting is show in 9 where w, x, y and z are integers. Matrices can be divided into certain columns of certain rows, this implementation is different to python 'matrix[x:y][w:z];' means that we take column w to z from rows x to y.

$$\begin{aligned} ts[x:y];\\ matrix[x:y]\\ ;matrix[x:y][w:z]; \end{aligned} \tag{9}$$

Extension again is similar to Python, but for PyDaSci is important to maintain the data structure, so only time series should be extended with other time series and matrices should be extended with other matrices to add new samples. The implementation of extension is shown in 10 and its symbol is '+'. Analysts can only add rows to matrices, there is no implementation to add new columns to matrices.

$$\begin{aligned} ts_1 + ts_2;\\ matrix_1 + matrix_2; \end{aligned} \tag{10}$$

To finalize this section is important to mention that comments are also supported in PyDaSci, comments are enclosed in $ symbols (e.g. $Comment$) and they can be single line comments or multi-line comments.

V. Future Work

There are many thing that can be still implemented in this new domain specific programming language. More forecasting methods for time series can be implemented, for example, machine learning algorithms as Support Vector Regression(SVR) or a neural network approach with Long Short-Term Memory architecture. Also, other clustering algorithms can be implemented to work with multivariate data as Birch clustering algorithm, OPTICS clustering or Bayssian-Gaussian mixture clustering algorithm. When talking about visualization there are also some features to improve as supporting visualization of the clustering results by applying Principal Component Analysis(PCA) to the matrix and coloring the samples according to the group it belongs to in a certain clustering algorithm.

VI. Conclusions

In this work, PyDaSci was introduced, a DSL for data analysis supporting forecasting and clustering algorithms. The main objective of this DSL is to provide programming abstraction for data analysts that are not expert programmers so they can easily and quickly implement their knowledge in this domain to analyze data without knowing much about programming. Abstraction is provided by limiting the attributes the programmer can control in the algorithms, making it simple to implement them. Due to the level of abstraction, better results can be obtained if other more complete programming languages are used, because more attributes of the forecasting and clustering algorithms can be modified. As it is an easy to use DSL, it manages the most important attributes to keep it simple for the programmer by sacrificing the complete control over certain algorithms. Despite this, results obtained with this DSL are good and useful for data analysis in many areas.

References

[1] S. Start, "Introduction to data analysis handbook migrant & amp: Seasonal head start technical assistance center academy for educational development," *Journal of Academic*, vol. 2, no. 3, pp. 6–8, 2006.

[2] T. W. Liao, "Clustering of time series data—a survey," *Pattern recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[3] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[4] M. Fowler, *Domain-specific languages*. Pearson Education, 2010.

[5] M. Weimer, T. Condie, R. Ramakrishnan *et al.*, "Machine learning in scalops, a higher order cloud computing language," in *NIPS 2011 Workshop on parallel and large-scale machine learning (BigLearn)*, vol. 9. Citeseer, 2011, pp. 389–396.

[6] A. de la Vega, D. García-Saiz, M. Zorrilla, and P. Sánchez, "Towards a dsl for educational data mining," in *International Symposium on Languages, Applications and Technologies*. Springer, 2015, pp. 79–90.

[7] O. González, R. Casallas, and D. Deridder, "Mmc-bpm: A domain-specific language for business processes analysis," in *International Conference on Business Information Systems*. Springer, 2009, pp. 157–168.

[8] S. Islam, S. Balasubramaniam, P. Goyal, M. Sati, and N. Goyal, "A domain specific language for clustering," in *International Conference on Distributed Computing and Internet Technology*. Springer, 2017, pp. 231–234.

[9] A. K. Sujeeth, H. Lee, K. J. Brown, T. Rompf, H. Chafi, M. Wu, A. R. Atreya, M. Odersky, and K. Olukotun, "Optiml: an implicitly parallel domain-specific language for machine learning," 2011.

[10] I. Portugal, P. Alencar, and D. Cowan, "A survey on domain-specific languages for machine learning in big data," *arXiv preprint arXiv:1602.07637*, 2016.

VII. Appendix

A. PyDaSci Grammar

1) Regular Expressions(Tokens): .

OPEN_PAREN : (
CLOSE_PAREN :)
OPEN_BRA : [
CLOSE_BRA :]
FLOAT : -?\d*\.\d+
INT : -?\d+
COLON : :
SEMI_COLON : ;
COMA : ,
ADD : +
METHOD : SARIMA|EXPSM
CLUSTERING : KMEANS|GAUSSIAN |AGGLOMERATIVE |SPECTRAL
BOOLEAN : False|True
TYPE_A : add|mul|additive|multiplicative
EQUAL : =
VAR : [a-z]+[a-zA-Z0-9]*
FORECAST : FORECAST
PREDICTION : PREDICT2|PREDICT
STATS : STATS
PRINT : PRINT
PLOT : PLOT
C_PLOT : C_PLOT
SENTENCE: '[^']*'

2) Productions: main : main assigment
main : assigment
assignment : STATS VAR SEMI_COLON
assignment : PLOT ts SENTENCE SEMI_COLON
assignment : PLOT VAR SENTENCE SEMI_COLON
assignment : C_PLOT plot_args SENTENCE SEMI_COLON
plot_args : plot_args VAR
plot_args : VAR
assignment : PRINT SENTENCE SEMI_COLON
assignment : PRINT VAR SEMI_COLON
assignment : VAR EQUAL program
assignment : VAR EQUAL ts SEMI_COLON
assignment : VAR EQUAL matrix SEMI_COLON
assignment : VAR EQUAL partition SEMI_COLON
assignment : VAR EQUAL extension SEMI_COLON
program : FORECAST VAR INT SEMI_COLON
program : METHOD OPEN_PAREN VAR args CLOSE_PAREN SEMI_COLON
program : CLUSTERING OPEN_PAREN VAR number CLOSE_PAREN SEMI_COLON
program : CLUSTERING OPEN_PAREN number CLOSE_PAREN SEMI_COLON
program : PREDICTION VAR VAR SEMI_COLON
args : pdq spdq
args : trend damped seasonal s_periods
pdq : OPEN_PAREN INT COMA INT COMA INT CLOSE_PAREN
spdq : OPEN_PAREN INT COMA INT COMA INT COMA INT CLOSE_PAREN
damped : BOOLEAN
trend : TYPE_A
seasonal : TYPE_A
s_periods : INT

number : INT
number : FLOAT
cn : COMA number cn
cn : ϵ
ts : OPEN_BRA number cn CLOSE_BRA
ts_c : COMA ts ts_c
ts_c : ϵ
matrix : OPEN_BRA ts ts_c CLOSE_BRA
partition : VAR OPEN_BRA number CLOSE_BRA
partition : VAR OPEN_BRA number COLON number CLOSE_BRA
partition : VAR OPEN_BRA number COLON number CLOSE_BRA OPEN_BRA number COLON number CLOSE_BRA
partition : VAR OPEN_BRA number CLOSE_BRA OPEN_BRA number COLON number CLOSE_BRA
extension : VAR ADD VAR