Ease the Process of Machine Learning with Dataflow

Tianyou Guo, Jun Xu[*], Xiaohui Yan[†], Jianpeng Hou,
Ping Li, Zhaohui Li, Jiafeng Guo, Xueqi Cheng
CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology,
Chinese Academy of Sciences
{guotianyou, houjianpeng, liping, lizhaohui}@software.ict.ac.cn,
{junxu, yanxiaohui, guojiafeng, cxq}@ict.ac.cn

ABSTRACT

Machine learning algorithms have become the key components in many big data applications. However, the full potential of machine learning is still far from been realized because using machine learning algorithms is hard, especially on distributed platforms such as Hadoop and Spark [8]. The key barriers come from not only the implementation of the algorithms themselves, but also the processing for applying them to real applications which often involve multiple steps and different algorithms. In this demo we present a general-purpose dataflow-based system for easing the process of applying machine learning algorithms to real world tasks. In the system, a learning task is formulated as a directed acyclic graph (DAG) in which each node represents an operation (e.g., a machine learning algorithm), and each edge represents the flow of the data from one node to its descendants. Graphical user interface is implemented for making users to create, configure, submit, and monitor a task in a drag-and-drop manner. Advantages of the system include 1) lowering the barriers of defining and executing machine learning tasks; 2) sharing and re-using the implementations of the algorithms, the task dataflow DAGs, and the (intermediate) experimental results; 3) seamlessly integrating the stand-alone algorithms as well as the distributed algorithms in one task. The system has been deployed as a machine learning service and can be access from the Internet.

Keywords

Machine learning process; dataflow; directed acyclic graph

1. INTRODUCTION

Machine learning has become the core of many big data applications such as information retrieval, question answering, and recommender system etc. To fulfill the increasing requirements on machine learning algorithms, a number of scalable machine learning libraries, including Apache Mahout [3] and Spark MLlib [4], have been developed and widely used. Despite the widespread impacts of the machine learning libraries, it is still difficult for ordinary users to use the machine learning in their applications. The barrier mainly comes from the complex process of using machine learning algorithms to solve a real-world task. A machine learning task usually consists of multiple steps including data preparation, feature extraction, model training, testing, and performance evaluation etc. The process could become more complicated if multiple learning algorithms and datasets are involved. For example, the user may want to use the topics as features in the task of document categorization. Thus, the process need to first train a topic model based on some dataset, and then feed the learned topics to a classification model. It has been widely recognized that constructing an appropriate process is crucial for the success of applying machine learning to real world application. Therefore, a platform that can help to ease the machine learning process will be of great help to users.

In this demo, we present a general-purpose machine learning system for lowering the barrier to applying machine learning algorithms. In the system, we consider the process of applying machine learning algorithms from the viewpoint of dataflow. Thus, the process can be formulated as a directed acyclic graph (DAG) in which the source data flow into the root nodes. Each node makes operations on the data, generates new data, and sends the generated data to its descendant nodes for conducting further operations. Finally, the results flow out from the leaf nodes.

The system consists of three major components: 1) A distributed machine learning library which implements popular machine learning algorithms as well as the algorithms for data pre/post-processing, format transformation, feature generation, performance evaluation etc. 2) A GUI-based machine learning studio which enable users to create, configure, submit, monitor, and share their machine learning process in a drag-and-drop manner. The algorithms in the machine learning library can be accessed in the studio. 3) A cloud service for executing the tasks. We build the service based on the open source big data platform of Hadoop and Spark. After receiving a task DAG from the GUI, each node will be automatically scheduled to run when all of its dependent data sources are ready.

The system offers several distinct advantages for applying machine learning to real tasks: 1) The dataflow formulation of machine learning tasks is quite intuitive and easy to understand. The GUI hides the unnecessary technical

[*]Corresponding author: Jun Xu
[†]Current affiliation: DiDi Research, Beijing

CIKM'16 October 24-28, 2016, Indianapolis, IN, USA
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4073-1/16/10.
DOI: http://dx.doi.org/10.1145/2983323.2983327

details of the algorithms (e.g., the complex command line) and helps user to focus on building the task process; 2) Users can upload and share their own data, algorithms, and tasks to other users; 3) It has the ability to use the stand-alone, Spark, and Map-Reduce algorithms in one DAG.

Several similar systems have been developed and released in enterprise and open source community. Mahout [3] and MLlib [4] are two distributed machine learning libraries developed with Hadoop Map-Reduce and Spark, respectively. A number of popular machine learning algorithms have been implemented in the libraries. To make these algorithms working together, people considers the application of machine learning as a workflow and several workflow schedulers have been developed, including the open source systems of Oozie [7], HUE [1], and Azkaban [2] etc. However, compare with the dataflow, the workflow cannot specify data dependencies between nodes, which increases the complexity of configuration. Microsoft has released Azure machine learning [5] in which a machine learning process is formalized as a dataflow.

2. MACHINE LEARNING PROCESS AS A DATAFLOW DAG

A typical application of the machine learning algorithms consists of several steps include gathering and preprocessing the data, extracting features, applying the training algorithm, and testing the performances of the trained model. From the viewpoint of data, the whole process can be viewed as the raw data flow into the processing pipeline. After a number of step-by-step operations on the data, the result data flow out the pipeline. The process could be more complex if multiple machine learning algorithms are involved.

Our system formulates the complex process of applying machine learning algorithms as a DAG of dataflow in which the data flow in the graph according to the directed edges. The data will be processed by the nodes it flows through. Each node consists of several input ports, output ports, and an operation. Each input port corresponds to an flow in data file and each output port corresponds to a data file that flows out. The operation is a (stand-alone or distributed) program that reads the input ports and writes the results to output ports. In our system, the operation of each node is implemented as a command line. A dataflow DAG may also contain some data nodes which represent the input data sources. Please note that a data node only have one output port. Figure 1 shows a typical dataflow DAG of applying logistic regression for identifying the SMS spam. Different colors are used to show the different status of the nodes: green for success, yellow for under executing, gray for waiting, and red for failed.

3. SYSTEM OVERVIEW

Figure 2 shows the architecture of the our system. The whole system consists of three parts: big data infrastructure for providing the foundation services, machine learning library for providing the core building blocks of the machine learning tasks, and machine learning studio for providing user-friendly GUI to lower the barrier of using machine learning.

• **Big data infrastructure**

Our system is built upon the open source big data system of Hadoop and Spark. All the data, machine learning algorithms, and other dependent information are stored in the distributed file system HDFS and data management system of Hive. A relational database system of MySQL is used for storing the metadata. Our system also depend on the distributed computational framework of Map-Reduce [6] and Spark [8]. All of the computational resources are managed with Yarn. Each of the submitted machine learning task (a dataflow DAG) is first converted to a workflow DAG and scheduled with the workflow scheduler system Oozie.

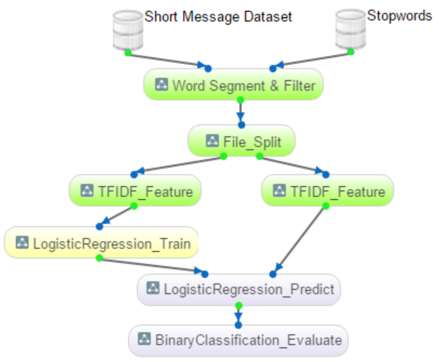


Figure 1: An example dataflow DAG.

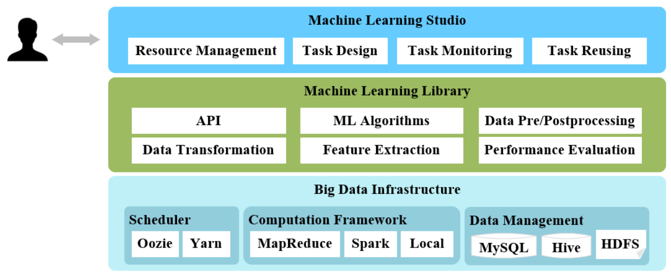


Figure 2: An overview of system architecture.

• **Machine learning library**

The machine learning library implements a number of popular machine learning algorithms (e.g., classification, topic modeling, graph processing, and information recommendation etc). For each algorithm, we implemented the distributed version on Spark as well as the standalone version because the standalone versions are usually more efficient than the distributed versions if the datasets are not big enough. Besides the core algorithms, the library also implements necessary modules for supporting the core algorithms including data pre/post-processing, data format transformation, feature extraction, and performance evaluation etc. All of the algorithms and modules can be called via both the command line and Java API. These algorithms constitute the core building blocks for users to define their machine learning tasks.

• **Machine learning studio**

The main goal of the machine learning studio is to provide a user-friendly GUI so that ordinary users can use the machine learning algorithms to solve their own problems easily. The machine learning studio is implemented as a web service and can be accessed via web browsers. It provides the follow main features:

(1) **Resource management**: All algorithms implemented in the machine learning library can be accessed from the

studio system. The system also provide a a number of data and tasks for showing how to use the algorithms to solve a problem. To construct a machine learning task, users may directly use the algorithms and data in the system. They can also upload their own data and algorithm packages. To upload an algorithm package, the user need to specify the format of the command line pattern string for running the algorithm. The string defines the program name, the input ports, the output ports, and the parameters settings. In this way, an uploaded algorithm can be run with different parameter settings. In a specified task dataflow DAG, the algorithm can be scheduled to run according to the command line pattern. After a machine learning task is submitted, it will be assigned a unique ID and stored in the task repository. Users can check and reuse the task in the future. They can also share the task to other users.

(2) **Task design**: To construct a machine learning task, a user may drag the algorithms and data sets (nodes) to the work panel, connect these nodes as a dataflow DAG, and set the parameters of all the nodes. If the users can find a similar task in our repository (in most cases), they can directly clone an existing task and make necessary modifications (add/remove nodes and edges, change parameters). By selecting a node in the work panel, the parameter setting panel will be shown in the right part of the page, which enables the users to set the specific parameter values for the corresponding algorithm in the task. After submitting a machine learning task, the studio will check the correctness of the dataflow DAG, generate the file paths of the temporal files, convert dataflow DAG to a work-flow DAG, and finally submit the work-flow DAG to Oozie for execution.

(3) **Task monitoring**: Users can monitor the progresses of a submitted task through the studio. During the execution of a task, different colors are used to indicate the status of the nodes: green for completed successfully, yellow for under running, red for completed with errors, and gray for waiting to execute. The results of a success node can be checked and downloaded via right clicking the corresponding output ports. The information printed to the standard out and standard error consoles can also be checked by right clicking the corresponding nodes. Through this way, users can know the status of a task and debug their algorithms and tasks if any error occurs.

(4) **Task Reusing**: An existing task can not only be used as templates for designing new tasks but also be reused for saving the execution time and system resource. Users may directly modify a completed task (e.g., modify the parameters of the nodes, add nodes and edges, or delete nodes and edges etc.) and resubmit the task. In the newly submitted task, only the influenced nodes are scheduled to execute and the results outputted by the uninfluenced nodes will be reused directly. For solving a real task, users usually need to tune their task dataflow DAG and parameters of the algorithms over and over. Task re-usage provides an effective mechanism to save the user's waiting time and resources.

4. ADVANTAGES

Our system offers the following advantages.

1) It is an easy-to-use and quite powerful system. The dataflow DAG formulation of the machine learning task is intuitive and easy to understand for ordinary users. Many unnecessary details are hidden. On the other hand, it still provides a lot of details (e.g., the parameters settings, the input/output ports etc.) for expert users.

2) The system seamlessly integrates the heterogeneous programs in one task. Since we used the HDFS for exchanging the information across different nodes, we have very few restriction on the form of the programs for the DAG nodes. The program corresponds to a node could be executed in stand-alone or distributed manner. It could be written with the programming language of C++, Java, Python, Perl or even the shell language.

3) The data, algorithms, and tasks in the system are highly reusable. Users can make use of the data and algorithms developed our library for constructing different machine learning tasks. The can also make use of the data and algorithms uploaded/shared by other users. A task can be cloned to construct similar tasks. Moreover, the intermediate results of an existing task can be reused by modifying and appending the task directly.

5. DEMO PLAN

We will present our system in the following aspects: (1) We will use a poster to give an overview of system architecture and briefly show the dataflow DAG formulation of the machine learning tasks and the system components. (2) We will show the audience how to use the system to complete a example machine learning task, including creating, configuring, submitting, and monitoring a task. (3) We will show the algorithms and datasets in the system. We will also show advanced functions of the system such as uploading new algorithms and datasets, sharing and reusing existing tasks etc. The audience will gain a deeper understanding on the system. (4) We will share our thoughts on the strengths and weakness of the system, and further discuss future work.

6. ACKNOWLEDGMENTS

This work was funded by the 973 Program of China under Grant No. 2014CB340401 and 2013CB329606, the 863 Program of China under Grant No. 2014AA015204, the National Natural Science Foundation of China (NSFC) under Grant No. 61502447, 61232010, 61425016, 61472401, and 61203298, and the Youth Innovation Promotion Association CAS under Grant No. 20144310 and 2016102.

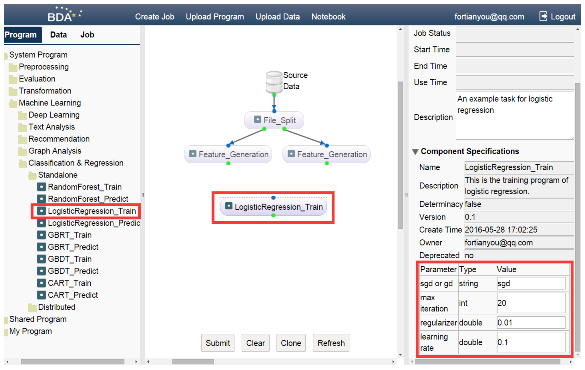
7. REFERENCES

[1] http://gethue.com/.
[2] https://azkaban.github.io/.
[3] https://mahout.apache.org/.
[4] http://spark.apache.org/mllib/.
[5] R. Barga, V. Fontama, and W. H. Tok. *Predictive Analytics with Microsoft Azure Machine Learning: Build and Deploy Actionable Solutions in Minutes*, chapter Introducing Microsoft Azure Machine Learning, pages 21–42. Apress, Berkeley, CA, 2014.
[6] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Operating Systems Design and Implementation*, 2004.
[7] M. Islam, A. K. Huang, M. Battisha, M. Chiang, S. Srinivasan, C. Peters, A. Neumann, and A. Abdelnur. Oozie: Towards a scalable workflow management system for hadoop. In *In SIGMOD Workshop on SWEET*, SWEET '12.
[8] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In *In 2nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, 2010.

APPENDIX

Our system can be accessed via http://159.226.40.104:18080/studio/ with a test account "**bdaict@hotmail.com**" and password "**bdaict**". For the best user experience, it is recommended to use Chrome.

As shown in the following figure, the users can create a machine learning task (a dataflow DAG) with the algorithms and data sets listed in the left panel of the page. They can choose to click the algorithms and data sets listed in the "Program" and "Data" panels. They can also click the "Job" panel, select an existing task, clone it, and make necessary modifications. The users can configure the task information and parameter values of each node in the right panel. The nodes in the task could corresponds to either a stand-alone Linux program or a distributed program running on Spark or Hadoop Map-Reduce.



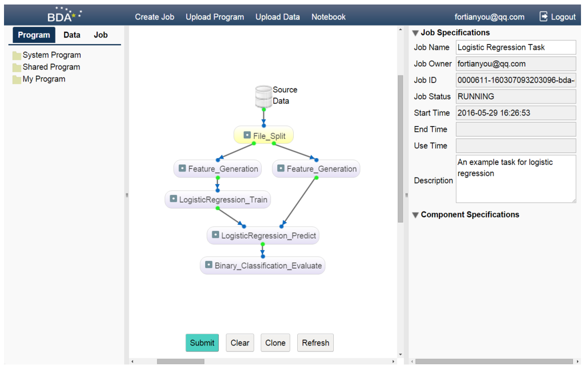
Job Status	
Start Time	
End Time	
Use Time	
Description	An example task for logistic regression

▼ Component Specifications

Name	LogisticRegression_Train
Description	This is the training program of logistic regression.
Determinacy	false
Version	0.1
Create Time	2016-05-28 17:02:25
Owner	fortianyou@qq.com
Deprecated	no

Parameter	Type	Value
sgd or gd	string	sgd
max iteration	int	20
regularizer	double	0.01
learning rate	double	0.1

The task is submitted to run on the cloud after clicking the "submit" button. The status of each node is indicated with different colors, as shown in the following figure.

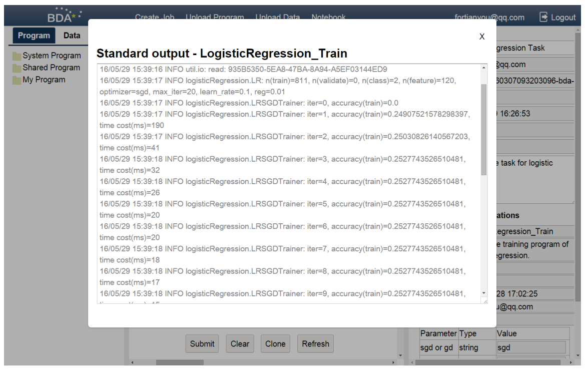


▼ Job Specifications

Job Name	Logistic Regression Task
Job Owner	fortianyou@qq.com
Job ID	0000611-160307093203096-bda-
Job Status	RUNNING
Start Time	2016-05-29 16:26:53
End Time	
Use Time	
Description	An example task for logistic regression

▼ Component Specifications

The users may check the outputs of a node by right clicking the corresponding output ports. The standard output and standard error information printed during the execution can be checked through right clicking the corresponding nodes and selects the menu "Show STDOUT" and "Show STDERR".

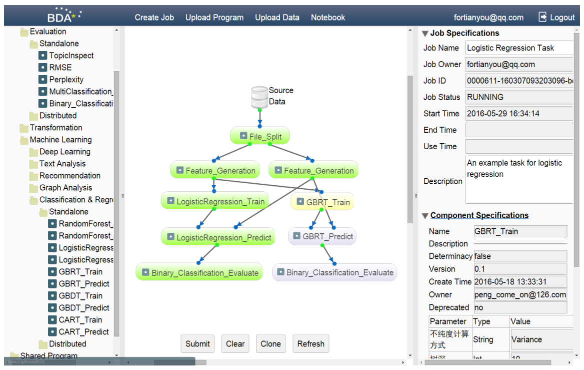


Standard output - LogisticRegression_Train

```
16/05/29 15:39:16 INFO util.io: read: 935B5350-5EA8-47BA-8A94-A5EF03144ED9
16/05/29 15:39:17 INFO logisticRegression.LR: n(train)=811, n(validate)=0, n(class)=2, n(feature)=120,
optimizer=sgd, max_iter=20, learn_rate=0.1, reg=0.01
16/05/29 15:39:17 INFO logisticRegression.LRSGDTrainer: iter=0, accuracy(train)=0.0
16/05/29 15:39:17 INFO logisticRegression.LRSGDTrainer: iter=1, accuracy(train)=0.24907521578298397,
time cost(ms)=190
16/05/29 15:39:17 INFO logisticRegression.LRSGDTrainer: iter=2, accuracy(train)=0.25030826140567203,
time cost(ms)=41
16/05/29 15:39:18 INFO logisticRegression.LRSGDTrainer: iter=3, accuracy(train)=0.2527743526510481,
time cost(ms)=32
16/05/29 15:39:18 INFO logisticRegression.LRSGDTrainer: iter=4, accuracy(train)=0.2527743526510481,
time cost(ms)=26
16/05/29 15:39:18 INFO logisticRegression.LRSGDTrainer: iter=5, accuracy(train)=0.2527743526510481,
time cost(ms)=20
16/05/29 15:39:18 INFO logisticRegression.LRSGDTrainer: iter=6, accuracy(train)=0.2527743526510481,
time cost(ms)=20
16/05/29 15:39:18 INFO logisticRegression.LRSGDTrainer: iter=7, accuracy(train)=0.2527743526510481,
time cost(ms)=18
16/05/29 15:39:18 INFO logisticRegression.LRSGDTrainer: iter=8, accuracy(train)=0.2527743526510481,
time cost(ms)=17
16/05/29 15:39:18 INFO logisticRegression.LRSGDTrainer: iter=9, accuracy(train)=0.2527743526510481,
```

Parameter	Type	Value
sgd or gd	string	sgd

A finished (either success or not) task can be further modified and resubmitted to run, as shown in the following figure. Our system will only schedule the influenced nodes to run. The outputs of uninfluenced nodes are directly reused to save the running time and system resources.



▼ Job Specifications

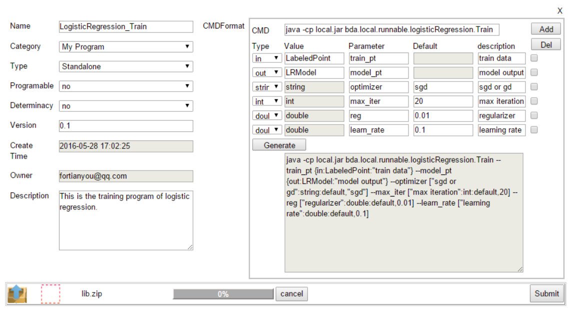
Job Name	Logistic Regression Task
Job Owner	fortianyou@qq.com
Job ID	0000611-160307093203096-b
Job Status	RUNNING
Start Time	2016-05-29 16:34:14
End Time	
Use Time	
Description	An example task for logistic regression

▼ Component Specifications

Name	GBRT_Train
Description	
Determinacy	false
Version	0.1
Create Time	2016-05-18 13:33:31
Owner	peng_come_on@126.com
Deprecated	no

Parameter	Type	Value
不纯度计算方式	String	Variance

The users can upload their own algorithm packages and data sets for creating their own tasks or shared with other users. By clicking the "upload program" button, the popup window allows the users to specify the necessary information of the algorithm package, including the name, the category, the description, and the command line pattern string etc, as shown in the following figure. The most important thing is to write the command line pattern string with the predefined format. It defined the input ports, output ports, and parameter settings of a node. We developed a tool in the panel for helping users to write the command line string patterns. By clicking the "upload data" button, users can upload a data set in the similar way as that of uploading a algorithms package.



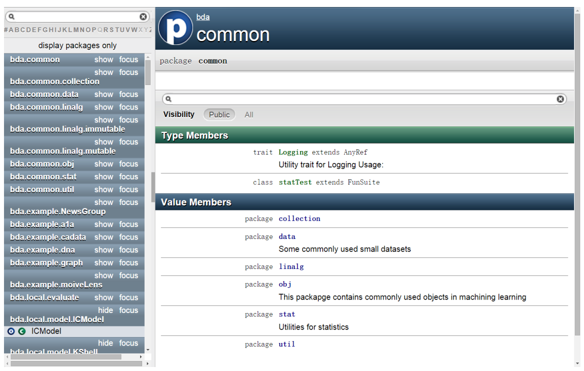
Name	LogisticRegression_Train
Category	My Program
Type	Standalone
Programable	no
Determinacy	no
Version	0.1
Create Time	2016-05-28 17:02:25
Owner	fortianyou@qq.com
Description	This is the training program of logistic regression.

CMD: java -cp local.jar bda.local.runnable.logisticRegression.Train

Type	Value	Parameter	Default	description
in	LabeledPoint	train_pt		train data
out	LRModel	model_pt		model output
strir	string	optimizer	sgd	sgd or gd
int	int	max_iter	20	max iteration
doul	double	reg	0.01	regularizer
doul	double	learn_rate	0.1	learning rate

```
java -cp local.jar bda.local.runnable.logisticRegression.Train --
train_pt {in:LabeledPoint:"train data"} --model_pt
{out:LRModel:"model output"} --optimizer ["sgd or
gd":string:default,"sgd"] --max_iter ["max iteration":int:default,20] --
reg ["regularizer":double:default,0.01] --learn_rate ["learning
rate":double:default,0.1]
```

All the algorithms in the machine learning library can also be called via Java API. The details can be found via http://159.226.40.104:18080/api/#package and the following figure shows the snippet.


#ABCDEFGHIJKLMNOPQRSTUVWXYZ
display packages only
bda.common show focus
bda.common.collection show focus
bda.common.data show focus
bda.common.linalg show focus
bda.common.linalg.immutable show focus
bda.common.linalg.mutable show focus
bda.common.obj show focus
bda.common.stat show focus
bda.common.util show focus
bda.example.NewsGroup show focus
bda.example.a1a show focus
bda.example.cadata show focus
bda.example.dna show focus
bda.example.graph show focus
bda.example.moiveLens show focus
bda.local.evaluate show focus
bda.local.model.ICModel hide focus
ICModel
bda.local.model.KShell hide focus
bda
common
Visibility
Public
All
Type Members
Value Members

package common

trait Logging extends AnyRef
Utility trait for Logging Usage:

class statTest extends FunSuite

package collection

package data
Some commonly used small datasets

package linalg

package obj
This packapge contains commonly used objects in machining learning

package stat
Utilities for statistics

package util