Towards a More Extensible Machine Learning Demonstration Tool

B. Thai, S. McNicholas, S.S. Shalamzari, P. Meng and J. Picone

The Neural Engineering Data Consortium, Temple University, Philadelphia, Pennsylvania, USA
{brianh.thai, shane.mcnicholas, somayeh.seifi.shalamzari, phuykong.meng, picone}@temple.edu

The Institute of Signal and Information Processing (ISIP) Machine Learning Demo (IMLD) is an educational tool designed to introduce the basics of machine learning through an easy-to-use graphical user interface (GUI) [1]. IMLD was first developed as part of a suite of Java applets in the late-1990's when Java applets were envisioned as the future of interactive computing [2]-[4]. IMLD visualizes the performance and results of various machine learning algorithms and techniques in a simple, intuitive GUI. A typical IMLD screen is shown in Figure 1. This paper introduces the recent enhancements to IMLD, which now hosts many new customizations, where its adaptability allows users to access more complex machine learning algorithms and the ability to implement their own algorithms as needed. IMLD provides benefits to the realm of medical signal processing as users are exposed to machine learning and the benefits that it provides in classifying data through various algorithms.

In recent years, IMLD has been converted to Python, where most machine learning research is now done [1]. The Java version of IMLD predated the development of packages such as Sklearn [5] and JMP [6] and contained its own implementations of many popular algorithms. Since then, as machine learning and artificial intelligence have become highly visible disciplines, the terminology has shifted, and the implementation of many standard algorithms has become more nuanced. Hence, there is a need to update IMLD to leverage the wide range of machine learning algorithms publicly available and to support parameter configurations that mirror the interfaces to sophisticated packages such as JMP. Hence, in this abstract, we introduce IMLD v2.0.0, which has a clearly defined interface to machine learning algorithms in Python and makes it easy to integrate new Python-based algorithms. It also supports a wider range of parameter settings that allow results to match popular packages such as Sklearn and JMP.

IMLD has a range of tools dedicated to teaching the basics of machine learning. Users can create two-dimensional data sets from either predefined or personal data. From here, a collection of standard machine learning approaches is available, such as both parametric and nonparametric algorithms. Data sets to be analyzed and other vital parameters are user-defined. Step-by-step algorithm computations are performed in a dialog box for the user to follow. IMLD produces rendered decision surfaces and error rates. This user interface structure has proved effective in teaching machine learning principles and remains largely unchanged over the years.

Aside from the user interface, another essential aspect of IMLD is data generation and handling. IMLD allows users to generate data using a variety of prestored models and drawing tools.

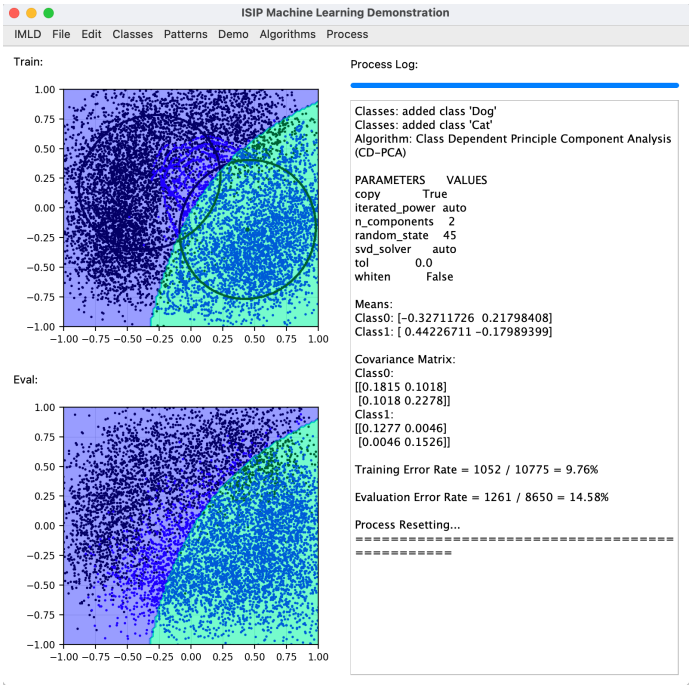


Figure 1. The IMLD v1.8.1 user interface

Standard data sets can be generated and customized through its drawing tools. Training and evaluation data are generated separately. Created data is exported to CSV files using a simple format that makes it easy to integrate IMLD into other data processing pipelines.

IMLD v2.0.0's software architecture has been redesigned so that the user interface and the algorithm classes are now completely separated. IMLD is still dependent on a variety of third-party libraries. IMLD continues to utilize PyQT5 [7] for its user interface, NumPy [8] for numerical computations, Sklearn/Scipy for implementing machine learning algorithms [5], and Matplotlib [9] for the visualization of data.

The new architecture, which is shown in Figure 2, has three major components: the graphical user interface, the library containing the algorithms (ml_tools), and the data handler. Most of IMLD's features and configurations are in a drop-down menu at the top of the user interface. These menus contain the elements for configuring classes and data, processing, and selecting algorithms. The algorithm implementations share a common interface through a library called as ml_tools. Previous computational algorithms for IMLD had been using non-conventional formatting and processing of the data. However, with ml_tools, the methods by which the data is being read and interpreted use a well-defined API. This library can be easily extended to include new algorithms by creating a script for the desired algorithm(s) along with the corresponding parameters through Python. A file template is provided as a guide for how implementing the necessary functions, parameters, etc. Through the API, users will have their functions linked into the application.

Due to this centralization of the algorithm code, we have expanded the algorithm selection and adapted terminology to use more modern names. This process presents complexities because there is a significant difference in the way packages like Sklearn and JMP refer to and implement standard algorithms such as Principal Components Analysis (PCA). For example, what was formerly called class-dependent PCA is now referred to as Quadratic Components Analysis algorithm (QDA). There is direct support for the following algorithms:

- Principal Components Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- K-Nearest Neighbor (KNN)
- Quadratic Components Analysis (QDA)
- Quadratic Linear Discriminant Analysis (QLDA)
- K-Means
- Naive Bayes (NB)
- Support Vector Machine (SVM)
- Multilayer Perceptron (MLP)
- Random Forests (RNF)

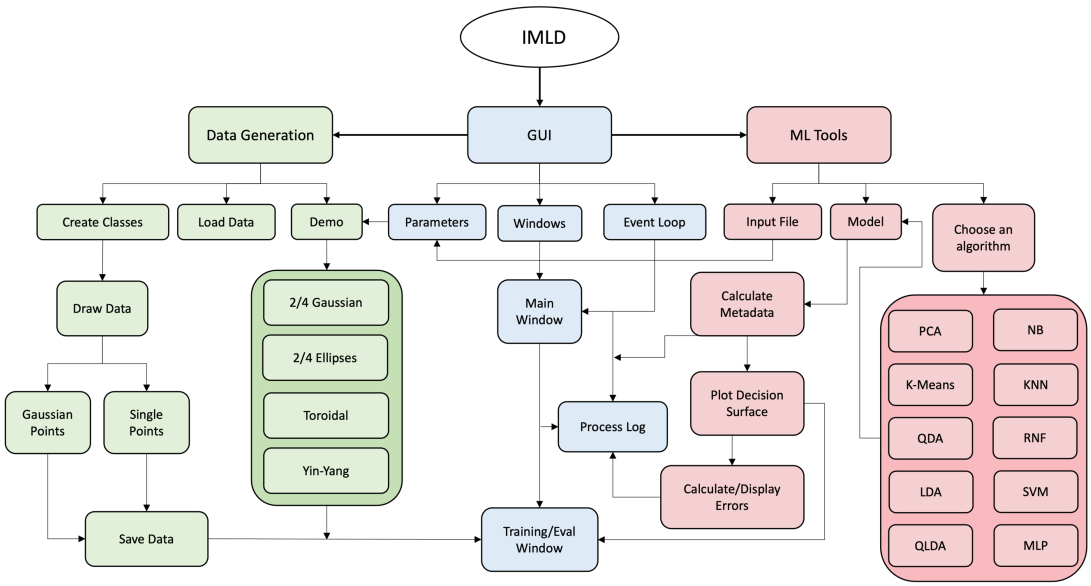


Figure 2. The IMLD software architecture

Since there are significant differences in the way Sklearn and JMP implement many of the same algorithms, the ml_tools library supports a wide range of parameter settings. This allows users to match results in either of these methods. For example, users can create a data set in IMLD, export it to JMP, and produce the exact same results on that data set using appropriate parameter settings. This was not possible in previous versions of IMLD, and matching results between Sklearn and JMP presents challenges. We have also expanded IMLD to support prior probabilities (probabilities that are assigned to incorporate initial beliefs about the outcome) and unbiased covariances (an estimate that asymptotically converges to the true value as the size of the data set grows) – two examples of parameter settings needed to match packages like JMP.

In the previous version of IMLD, values like parameters and settings were hardcoded directly into the program. For example, the drop-down menus and parameters were embedded within the user interface code, making it difficult for users to add their own choices. By exploiting Python's excellent data-driven programming support, algorithms can be added without modifying the code. Menu choices are driven from descriptions in external parameter files that are loaded at run-time. This makes it much easier for users to customize the tool.

IMLD is a tool that allows users to easily explore machine learning algorithms on predefined and user-defined data sets. Its unique visualization of the machine learning process makes it an ideal teaching tool. It has been used by a machine learning class we have been teaching since the late 1990s (*https://www.isip.piconepress.com/courses/temple/ece_8527/*). The source code is available from the course website at: *https://www.isip.piconepress.com/courses/temple/ece_8527/resources/imld/*. A detailed user manual demonstrating the use of the tool and instructional videos are also available. A demonstration will be provided at the symposium.

ACKNOWLEDGEMENTS

The development of IMLD has been supported by many grants over the past three decades, including grants from the National Science Foundation, the National Institutes of Health, and the Temple University Office of Research. The material presented in this abstract was supported by the Temple University College of Engineering's Summer Research Experience for Undergraduates program. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of these sponsors.

REFERENCES

[1] T. Cap, A. Kreitzer, M. Miranda, D. Vadimsky, and J. Picone, "IMLD: A Python-Based Interactive Machine Learning Demonstration," in *Proceedings of the IEEE Signal Processing in Medicine and Biology Symposium* (SPMB), I. Obeid, I. Selesnick, and J. Picone, Eds., Philadelphia, Pennsylvania, USA, 2021, pp. 1–4. doi: *10.1109/SPMB52430.2021.9672265*.

[2] J. Shaffer, J. Hamaker, and J. Picone, "Visualization of signal processing concepts," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Seattle, Washington, USA, May 1998, pp. 1853–1856. doi: *10.1109/ICASSP.1998.681824*.

[3] D. May and J. Picone, "The ISIP Pattern Recognition Applet," Institute for Signal and Information Processing, College of Engineering, Mississippi State University, 2002. [Online]. Available: *https://www.isip.piconepress.com/projects/speech/software/demonstrations/applets/util/pattern_recognition/current/*. [Accessed: 05-Jul-2021].

[4] J. Picone, R. Duncan, and J. Hamaker, "Internet-Accessible Speech Recognition Technology," in O'Reilly Open Source Convention, 2001. url: *http://www.isip.piconepress.com/publications/conference_presentations/2001/oscon/software/*.

[5] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011, doi: *10.5555/1953048.2078195*.

[6] JMP®, SAS Institute Inc., Cary, North Carolina, USA, 1989 – 2023. Available: *jmp.com*.

[7] PyQT, "PyQt Reference Guide," 2023, [Online]. Available: *http://www.riverbankcomputing.com/static/Docs/PyQt5/html/index.html*.

[8] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. *Array programming with NumPy*. Nature 585, 357–362 (2020). doi: *10.1038/s41586-020-2649-2*.

[9] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

Towards a More Extensible Machine Learning Demonstration Tool

B. Thai, S. McNicholas, S. S. Shalamzari, P. Meng and J. Picone

The Neural Engineering Data Consortium, Temple University

NEURAL ENGINEERING DATA CONSORTIUM
www.nedcdata.org

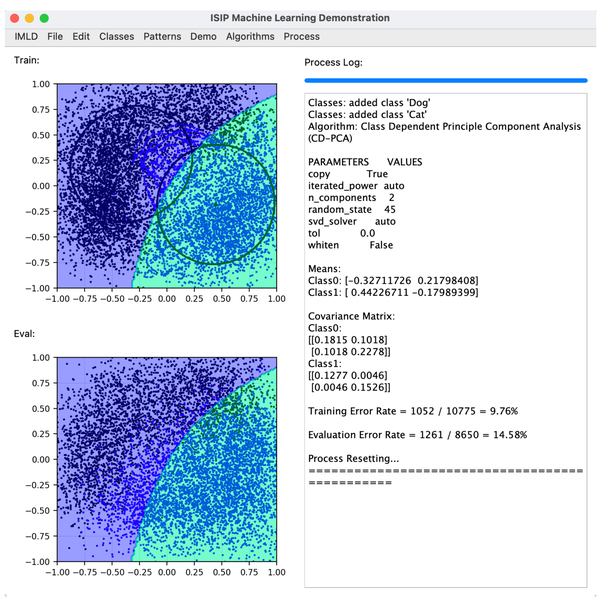
College of Engineering
Temple University

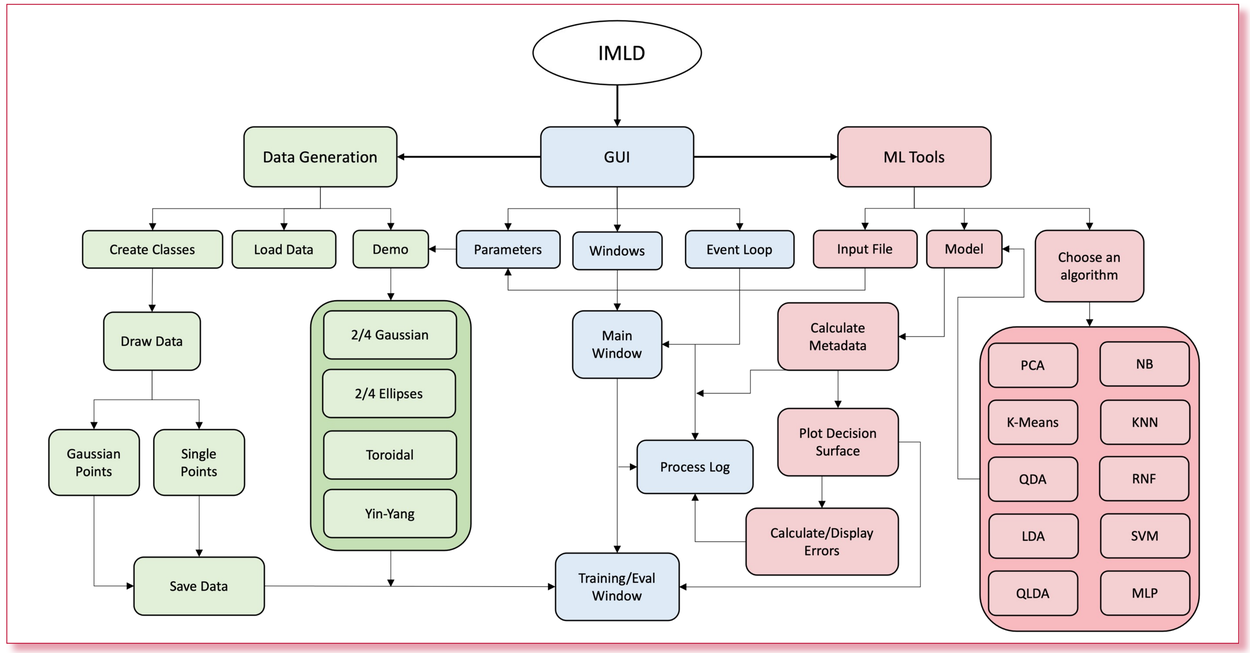
Abstract

- The ISIP Machine Learning Demo (IMLD) was originally developed in the 1990's as a Java applet to demonstrate key machine learning (ML) concepts.
- Since then, it has evolved into an open source Python application based on PyQt that encapsulates many modern ML algorithms. Data sets can be created, evaluated and visualized.
- IMLD displays decision surfaces, error rates and model parameters in an associated dialog window.
- In the most recent version (v2.0.0), we are including a new library, known as ML Tools, that supports of the standard SKlearn ML algorithms and makes adding new algorithms relatively easy.
- In this abstract, we review the new API and demonstrate how to integrate custom algorithms.

The ISIP Machine Learning Demo (IMLD)

- IMLD is an educational tool designed to introduce the basics of machine learning through an easy-to-use graphical user interface (GUI).
- In recent years, IMLD was converted to Python, and supports the popular ML library SKlearn. Its visualization software is based on PyQT.
- IMLD has a range of tools to teach the basics of machine learning. Predefined datasets are available internally, and it supports importing of custom data.
- Supervised/unsupervised algorithms and parametric/nonparametric models are supported.
- IMLD displays data and decision surfaces, computes error rates, and exports trained models so that these models can be integrated with other Python code.
- IMLD has proven to be an effective tool to exposing beginners to machine learning principles and has been in use in undergraduate and graduate machine learning classes since the 1990's.





Modifications to IMLD

- The Java version of IMLD predated the development of packages such as Sklearn and JMP and had its own implementations of various machine learning algorithms. As the field of machine learning has changed and the terminology has changed, modifications had to be made to mirror the results of sophisticated packages like JMP
- IMLD now supports a wider range of parameter settings that allow the tool to be configured to match Sklearn and JMP. For example, there are subtle differences in the way Sklearn and JMP compute covariances. IMLD can now match results produced by either tool.
- Another major change to IMLD is the structure of the application. The new architecture has three major components: the GUI, the ML Tools library, and data generation. This has simplified the process of adding a new algorithm to IMLD, since the machine learning code is now isolated in the ML Tools library.
- Finally, IMLD was modified by removing the hardcoded parameters and settings within the GUI. Menu parameters are now loaded from an external parameter file that is loaded at run time, allowing for more customizable options.

ML Tools

- ML Tools is a library that contains the machine learning algorithms that IMLD implements in the GUI.
- Previous computational algorithms for IMLD had been using non-conventional formatting and processing of the data sets.
- Through ML Tools, the method through which the datasets are being analyzed are done through a well-defined API, which allows for consistency amongst the results regardless of packages.
- With ML Tools, the user interface (GUI) is completely separated from the classes containing the machine learning algorithms.
- For each algorithm within ML Tools, key results from training, such as labels generated during unsupervised training, are accessible. This allow both supervised and unsupervised algorithms to be supported.
- ML Tools uses an innovative method of constructing a dictionary containing the collection of algorithms. This makes adding new algorithms very easy.
- IMLD is now essentially a presentation system (GUI).

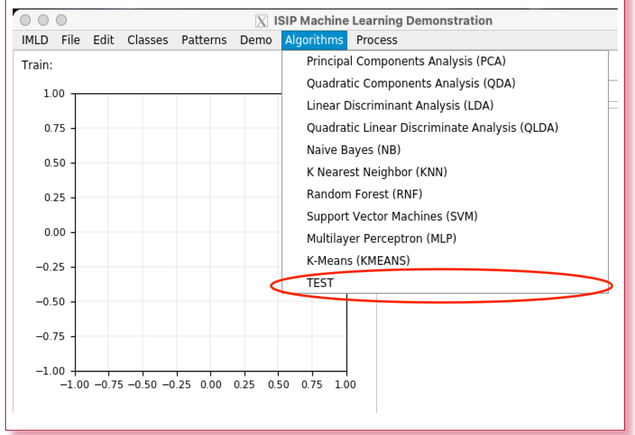
```
# define the algorithm name
#
QDA_NAME = "QDA"

# the parameter block for QDA looks like this:
#
#   defaultdict(<class 'dict'>, {
#    'name': 'QDA',
#     'params': {
#       'name': 'QDA',
#       'prior': 'ml',
#       'ctype': 'full',
#      'center': 'None',
#       'scale': 'biased'
#     }
#   })
#
# Define the keys for these parameters.
#
QDA_PRM_KEY_NAME = ALG_PRM_KEY_NAME
QDA_PRM_KEY_PARAM = ALG_PRM_KEY_PARAM
QDA_PRM_KEY_PRIOR = ALG_PRM_KEY_PRIOR
QDA_PRM_KEY_CTYPE = nct.PRM_CTYPE
QDA_PRM_KEY_CENTER = nct.PRM_CENTER
QDA_PRM_KEY_SCALE = nct.PRM_SCALE
QDA_PRM_KEY_COMP ='n_components'
```

```
# define variables to configure the machine learning algorithms
#
ALGS = {PCA_NAME: PCA(), LDA_NAME:LDA(), QDA_NAME:QDA(),
        QLDA_NAME: QLDA(), NB_NAME:NB(), KNN_NAME:KNN(),
        RNF_NAME:RNF(), SVM_NAME:SVM(), KMEANS_NAME:KMEANS(),
        MLP_NAME:MLP()}
```

Adding Additional Algorithms

- Previously, algorithms were implemented within the IMLD driver program, which restricted a user's ability to customize the tool.The software architecture has been redesigned so that the user interface and the algorithm classes are now completely separated.
- IMLD is still dependent on a variety of third-party libraries. IMLD continues to utilize PyQT5 for its user interface, NumPy for numerical computations, Sklearn/Scipy for implementing machine learning algorithms, and Matplotlib for the visualization of data.
- To allow for users to create and add their own algorithms, a Python template was made for user-defined algorithms. The template specifies formatting, variable naming, and algorithm naming.
- Once the templates are completed, the GUI reads from the ML Tools library and the input algorithm file, build the algorithms, and dynamically add them to IMLD.
- An example of the addition of a new algorithm, name TEST, is shown here:



Summary

- IMLD is a tool that allows users to explore machine learning algorithms on predefined and user-defined data sets. Its unique visualization of the machine learning process makes it an ideal teaching tool.
- Users also have the ability to implement their own algorithms using the templates provided, and can now evolve IMLD as packages like Sklearn and JMP add new algorithms and features.
- Future research will include integrating transformer/self attention algorithms and quantum computing based machine learning.

Acknowledgements

The development of IMLD has been supported by many grants over the past three decades, including grants from the National Science Foundation, the National Institutes of Health, and the Temple University Office of Research. The material presented in this abstract was supported by the Temple University College of Engineering's Summer Research Experience for Undergraduates program. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of these sponsors.

To learn more about our resources, please use this URL:
https://isip.piconepress.com/projects/imld