Document-centered Data Analysis Workflows with integrated open Mathematica Packages for Curve Fitting, Clustering and Machine Learning

Abstract

Curve fitting, data smoothing, clustering and machine learning are important computational optimization techniques for scientific data analysis. A set of integrated packages (*CIP*) based on the Mathematica computing platform alleviate and unify these kinds of data analysis workflows in a document-centered environment with a flexible combination of high-level code, (graphical) results and customizable style elements. The open *CIP* packages itself may be easily extended or tailored for specific needs.

Keywords

non-linear curve fitting; clustering; cluster analysis; machine learning; multiple linear regression; multiple polynomial regression; feed-forward neural network; perceptron; support vector machine; classification; regression; model construction; training/test set selection; hyper-parameter optimization; feature relevance; data analysis workflow; document-centered interface; rich-client; pipelining-workflow; open libraries; open-source.

Introduction

The interplay between experiment and theory is a key driving force of modern science with data analysis as a connector in between: Its techniques advise possibilities of theoretical model extraction as well as model testing with experimental data [1-3]. Theory-driven research areas like physical chemistry or biophysics do often experimentally study a scientific quantity of interest (y) in dependence of another quantity (x). This dependency is theoretically described by a model function y = f(x) where often only the structural form of f(x) is known (e.g. that f is a straight line) but not the values of its parameters (e.g. the values of a and b of the straight line y = f(x) = a + bx). Curve fitting methods allow estimating the parameter values of a known structural model function on the basis of experimental data with errors that follow a specific statistical distribution (which often is the normal distribution). Pure data smoothing techniques may be applied if even the structural form of y = f(x) is unknown. With increasing complexity of the natural system under investigation a quantitative theoretical treatment becomes more and more difficult, e.g. contemporary molecular science fails to predict biological activities of new molecular entities or the properties of new material's compositions [4-9]. To achieve at least an approximate quantitative relationship between the experimentally determined properties and an adequate set of descriptive system features of the form property = f(feature 1, feature 2, feature 3, ...) a model function f may be solely constructed on the basis of feature/property pairs: A task that is at heart of machine learning. The distribution of the descriptive features within the feature space itself may be investigated by clustering techniques, e.g.

to select representatives for training and test sets which are mandatory for the validation of machine learning results.

For computational data analysis workflows that tackle curve fitting, data smoothing, clustering or machine learning different application types have evolved and are currently in use in academic and industrial practice: Rich-client systems [10], pipelining-workflow applications [11] and scripted software libraries [12]. Each application type has its own strengths and weaknesses: Rich-clients are most comfortable to use and do not require data analysts with specific programming or technical skills so they are widely used in scientific research and development (where data analysis is often part of electronic laboratory notebooks). On the other hand rich-client software is monolithic and rigid when it comes to customized workflow modifications - changing or extending is time-consuming, expensive and requires highly skilled computer scientists. To alleviate customization pipelining-workflow applications became popular in recent years: They provide a computational rich-client framework that encapsulates specific data analysis tasks in modular units that each can be graphically connected in a LEGO™-like manner to create the desired workflows in a more flexible manner. Programming skills of users are still not required but a deeper insight and training is necessary in addition with challenges concerning the management of (the plethora of) result files and their documentation. Last but not least pure software libraries are the most flexible basis for data analysis: Successive library calls and corresponding data structures are top-down composed with a scripting language to construct the tailored workflow in need. But this requires specifically skilled computer scientists like bioinformaticians which are often not available or simply too expensive.

This work suggests a hybrid application type of data analysis workflows for curve fitting, data smoothing, clustering or machine learning. For user interaction a document-centered interface (DCI) is chosen which is available on mathematical computing platforms [13]. A DCI combines software code, textual features (fonts etc.), organizational styles (chapters, sections, paragraphs etc.) or graphical results up to GUI widgets within one single document/file which may be stepwise or completely evaluated as well as manipulated within the computing platform. The data analysis tasks mentioned above are tackled with specific high-level function libraries that provide abstracted methods with comprehensive data structures (where the subtleties are hidden but are all accessible) that are developed on the basis of the platform's programming languages and its built-in algorithms. The application of these highly abstracted libraries within a document-centered user interface allows the setup of data analysis workflows with a productive combination of comfort and flexibility as well as to address the requirements of proper information management, documentation and reproducibility. Each workflow results in one single document file that contains all code, graphics and annotations. If necessary a workflow may be boosted with the complete set of the platform's functions offering a plethora of I/O, manipulation and calculation methods.

For this work the Mathematica system was chosen as a computing platform [14]. The specifically developed high-level data analysis libraries (*CIP* [15], see next section) are open-source and may themselves be used as a starting point for customized and tailored extensions.

Software and Methods

CIP (Computational Intelligence Packages) consists of high-level function libraries which are developed on top of the Mathematica computing platform. *CIP* utilizes the platform’s built-in

programming language, optimization algorithms and graphical capabilities to support scientific data analysis workflows for (non-linear) curve fitting and data smoothing (with cubic splines), clustering (k-medoids, ART-2a) and machine learning (multiple linear/polynomial regression, 3-layer feed-forward perceptron-type neural networks and support vector machines). For machine learning *CIP* supports classification as well as regression tasks. In addition the packages provide several heuristics for the selection of training and test data, hyper-parameter optimization or methods to estimate the relevance of data input components. A number of useful auxiliary functions and experimental data for testing purposes complement the offer.

For ease of use *CIP*-based workflows follow an intuitive and unified Get-Fit-Show/Calculate scheme: With Get methods data are retrieved from the *ExperimentalData* package, simulated with the *CalculatedData* package or may be provided from any source the computing platform is able to communicate with. The data are then submitted to a Fit method of the *CurveFit*, *Cluster*, *MLR*, *MPR*, *SVM* or *Perceptron* packages to perform a corresponding optimization procedure. The result of a Fit method is a comprehensive info data structure, e.g. a curveFitInfo, clusterInfo, mlrInfo, mprInfo, svmInfo or perceptronInfo. This info data structure may then be passed to Show methods for multiple evaluation purposes like visual inspection of the goodness of fit or to Calculate methods for model related calculations. Similar functions of different packages are denoted in a similar manner for convenient applicability and intuitive method changes. Function signatures do mainly contain only method-dependent structural hyper-parameters (like the number of hidden neurons for perceptrons or the kernel function for support vector machines) whereas technical control parameters (like the maximum number of iterations) may be changed via options if necessary. Some *CIP* packages do only perform auxiliary tasks like the *Graphics* package that provides standardized 2D and 3D diagrams.

The *CIP* design goals were neither maximum speed nor minimum memory consumption but a unified definition of robust high-level functions for scientific data analysis workflows. Thus *CIP* is not a generally optimized and maximum efficient library for any scientific application although it may and is practically utilized in many operational areas (see individual package comments below). Since *CIP* is LGPL open-source the library may be used as a starting point for customized and tailored extensions, e.g. an implementation of a multicore processor support to increase computational speed (many *CIP* operations are ideally suited for parallel operation).

CIP consists of 11 integrated packages:

Utility - basic package that collects several general functions (like GetMeanSquaredError which is used by all machine learning related packages) to decrease redundant code.

ExperimentalData - provides test data.

DataTransformation - performs internal data transformations for different purposes, e.g. all data that are passed to a machine learning method are scaled before a Fit operation.

Graphics - tailors Mathematica's built-in graphical functions for diagrams and graphical representations.

CalculatedData - complements the *ExperimentalData* package with methods for the generation of simulated data like normally distributed xy-error data around a defined function, e.g. for the test of curve fitting workflows.

CurveFit - tailors Mathematica's built-in curve fitting method (NonlinearModelFit [17]) for least-squares minimization and adds a smoothing cubic splines support [18].

Cluster - tailors Mathematica's built-in (k-medoids) FindClusters method [21] for clustering purposes and adds an Adaptive Resonance Theory (ART-2a) support [22, 23].

MLR/MPR - tailor Mathematica's built in Fit method [27] for multiple linear or polynomial regressions (MLR/MPR).

Perceptron - provides optimization algorithms for three-layer feed-forward perceptron-type neural networks [4, 28, 29]. It utilizes Mathematica's FindMinimum (ConjugateGradient [30]) or NMinimize (DifferentialEvolution [31-34]) methods for unconstrained minimization tasks. The package also provides a backpropagation-plus-momentum minimization and a genetic-algorithm-based minimization [35]. Although the quality of the minimization algorithms is state-of-the-art the specific calculation setup contains non-optimum redundancies that decrease performance and increase memory consumption.

SVM - provides constrained optimization algorithms for support vector machines (SVM). It utilizes Mathematica's FindMaximum (InteriorPoint [36, 37]) or NMaximize (DifferentialEvolution [31-34]) methods for constrained optimization tasks [38]. Although these algorithms are robust they do not exploit any specifics of the support vector objective function to increase convergence speed etc.

For deployment to the Mathematica computing platform the *CIP* folder [15] with the 11 package (.m) files should be located in Mathematica's "<Mathematica Installation Location>/AddOns/Applications" directory.

Application Example

A *CIP*/Mathematica -based data analysis workflow [39] is discussed by screenshots of the corresponding document-centered interface that describes the step-by-step construction of a benign/malignant tumor class predictor for the Wisconsin Diagnostic Breast Cancer (WDBC) data [40].

The WDBC classification data set in question consists of 569 input/output (I/O) pairs where each input is mapped onto one of two output classes. Every I/O pair refers to a single patient. The 30 components/features of each input are real-valued quantities that are computed from a digitally scanned image of a fine needle aspirate of a breast mass and describe characteristics of the extracted cell nuclei present in the digital image. The two output components code two classes where class 1 denotes the diagnosis of a benign tumor and class 2 the diagnosis of a malignant tumor.

The complete data analysis workflow is sketched in Figure 1: It starts with 2 descriptive sections (System Requirement and Task Description, both collapsed) that provide initial information. The data itself are retrieved from the *CIP ExperimentalData* package and stored in the wdbcDataSet variable (see expanded section Data Source). All following data analysis sections are collapsed for overview and discussed subsequently.

Inspection of the data set (with named properties, see Figure 2) shows that the number of benign class 1 tumor samples considerably exceeds the number of malignant class 2 tumor samples – an asymmetry that is unwanted in general.

A classification task should always be tackled by unsupervised learning first: Unsupervised learning is comparatively fast (seconds to minutes on today's workstation computers for moderate data set sizes like the one under consideration) and if it succeeds there is no need for the more subtle and slower supervised learning methods. For construction of a class predictor by unsupervised learning the FitCluster method of the *Cluster* package is invoked (see Figure 3). This methods extracts the inputs of all I/O pairs of the data set and partitions them in an unsupervised manner (by the default k-medoids method) into the desired number of clusters equal to the number of classes (which is 2 in this case). Then it determines the cluster centroids and evaluates the class members of each cluster. Finally each centroid is attributed the class of the majority of class members in its cluster. Classification of new inputs means to compute the Euclidian distance to both determined centroids with the smaller distance winning for the class to assign. The result of the whole unsupervised learning process is coded in a resulting clusterInfo data structure. This data structure may be used to evaluate the predictive success for the WDBC data of the unsupervised learning approach with an appropriate Show method (ShowClusterSingleClassification in this case, see Figure 3). 85% correct classifications indicate that the input features of both classes do not form clearly separated clusters in the input space but overlap. A closer look at the class analysis histograms reveals that the correct benign class 1 tumor predictions are nearly perfect (near 100%) but the malignant class 2 tumor predictions comparatively poor (61% - but note from the wrong classification distribution that if a tumor was predicted to be malignant this information is quite reliable). Thus unsupervised learning is not able to construct a satisfactory predictor for the data in question and the application of supervised learning methods is advised.

A good start for supervised machine learning is the application of a linear MLR (multiple linear regression) method: A linear method is again fast and not prone to overfitting (a problem that is encountered in a minute) – but on the other hand very limited due to the restriction of only linear decision hyper planes. The use of the FitMlr method leads to a resulting mlrInfo data structure that encapsulates the linear model (see Figure 4, note the complete analogy to the use of the FitCluster method above). Again the machine learning success may be evaluated by an appropriate Show method (now ShowMlrSingleClassification): With an overall 96% correct classifications a considerable improvement in comparison to the unsupervised learning result could be achieved. While the benign class 1 tumor predictions are still nearly perfect the malignant class 2 tumor predictions were raised from 61% to over 90%. Another interesting issue may be analyzed: How relevant is each of the 30 input features for the benign/malignant classification in question? Or: What is the smallest subset of input features that achieves a predictive success comparable to the full input feature set? *CIP* provides heuristical methods [41] to tackle these (extremely difficult) questions (since the number of input feature combinations lead to a "combinatorial explosion"). The GetMlrInputInclusionClass method successively checks input features for their relevance to the desired classification task (see Figure 5). The result is kept in the mlrInputComponentRelevanceListForClassification variable and may be displayed with another Show method (ShowMlrInputRelevanceClass). It becomes obvious that only 4 out of 30 input features are sufficient to obtain a predictive success comparable to the full feature set (where feature 28 is most, feature 21 second, feature 22 third and feature 24 fourth relevant. Note that the detected set of 4 input features is not necessarily the true optimum feature set of size 4: A heuristical approach does not guarantee to arrive at any true optimum. The optimum feature set issue could be tackled by evolutionary optimization strategies but these strategies deserve extraordinary computational power far beyond simple heuristical approaches). Thus a considerably reduced data set can be obtained by deleting the other 26 non-relevant features (which

is performed by the IncludeInputComponentsOfDataSet method). As a last step it is advised to validate the model's predictive power since a good predictability on the full data set does not necessarily mean a good performance on new data: This is known as the already mentioned problem of overfitting where the obtained model acts as a mere "look-up table" for the training data but has no generalization abilities for data unknown to the construction/training process. To validate the model the reduced WDBC data set (with now only 4 input features) is split into a training and a test set of equal size (the trainingFraction is 0.5, see Figure 6). *CIP* again provides specific heuristical methods [41] for training/test set partitioning which use optimized cluster representatives for training that guarantee the training and test I/O pairs to possess a similar spatial distribution in the input feature space (see method GetMlrTrainOptimization in Figure 6. Note that a simple random partitioning does often not lead to similar spatial distributions, e.g. if inputs are distributed with large differences in density). The best training/test set combination has a predictive success of over 96% correct classifications for both sets so overfitting can be ruled out (which is a priori expected for a restricted linear method that is not prone to overfitting in general).

To possibly arrive at a benign/malignant tumor predictor with even improved predictive power the sound grounds of linearity must be left to enter the realm of non-linearity. The application of non-linear machine learning methods is more subtle, contains many possible pitfalls and deserves a increased computational power. Proceeding from linear to polynomial decision surfaces with the *MPR* package achieves the most humble non-linear extension: A parabolic approach with a polynomial degree of 2 (note that every non-linear machine learning method needs the a priori definition of so called hyper-parameters that guide its learning behavior – and unfortunately there is in general no way to theoretically estimate these parameters in advance). The parabolic approach (again completely analogous to the MLR approach before) leads to a 100% correct classifications for the full WDBC data set: A suspiciously perfect result that indicates overfitting (see Figure 7). If the full data set is partitioned in training and test data of equal size as before overfitting is validated by a poor result on the test set with only 80% correct predictions (which is even below the unsupervised learning result above). A parabolic decision surface above the 30 feature input space is already to flexible and just leads to a "look-up table" without satisfactory generalization abilities. To decrease flexibility a reduced data set with again only 4 input features is constructed: In this case this leads to the same subset of 4 input features as before and after training/test set partitioning to a classification result comparable to the linear MLR approach. A more non-linear cubic approach with a polynomial degree of 3 on a 4-input-features reduced data set (again with the same subset as a result) and training/test set partitioning finally arrives at nearly 97% correct classifications on both sets which is a small improvement especially for the malignant class 2 predictions (see Figure 8).

Very popular families of deeply non-linear machine learning methods are neural networks and support vector machines (SVM). *CIP* provides 3-layer feed-forward (3L-FF) neural networks (perceptrons) and different kernel functions for SVM construction. A perceptron approach with 3 hidden neurons (the crucial hyper-parameter of 3L-FF neural networks) again on a 4-input-features reduced data set and training/test set partitioning reaches nearly 98% correct classifications on both sets (see Figure 9, note that the minimal set of 4 input features now clearly deviates from those before – there is nothing like a general best subset of input features). An increase of the number of hidden neurons leads to an increased learning capacity but also increases the networks proneness to overfitting. A 5-hidden-neuron perceptron leads to over 98% correct classifications and a 7-hidden-neuron-perceptron finally comes close to a 99% success on both sets. A further increase of the

number of hidden neurons does not lead to improved predictability. An appropriate SVM approach would lead to a comparable classification success.

In summary the data analysis workflow continued from unsupervised to supervised and from linear to increasingly non-linear machine learning methods to arrive at a convincing predictor with a nearly 99% validated success rate. Due to the definition of the *CIP* libraries all single different analysis sections are genuinely similar in design and succession of commands. The DCI computing environment allowed a comfortable development of a clearly structured and comprehensive document ready for export to other formats (like PDF) or presentation purposes as well as a blueprint for other corresponding data analysis workflows.

Summary

CIP supports data analysis workflows from curve fitting and data smoothing to clustering and machine learning in a document-centered environment on the Mathematica computing platform. The provided high-level functions use corresponding signatures for the various methods available that alleviate method change and analogue data analysis workflows. In addition to the Fit methods *CIP* offers heuristic procedures for additional tasks (like hyper-parameter optimization or training and test set partitioning) as well as standardized diagrams for graphical inspection and methods for model based calculations. The open packages may be easily tailored for specific needs or optimized for parallel processing and improved performance.

References

[1] W. C. Hamilton, Statistics in the Physical Sciences, New York 1964, Ronald Press.

[2] P. Bevington, D. K. Robinson, Data Reduction and Error Analysis for the Physical Sciences, New York 2002, McGraw-Hill, 3rd Edition.

[3] S. Brandt, Data Analysis: Statistical and Computational Methods for Scientists and Engineers, New York 1998, Springer, 3rd Edition.

[4] J. Zupan, J. Gasteiger, Neural Networks in Chemistry and Drug Design, Weinheim 1999, Wiley-VCH.

[5] A. R. Leach, Molecular Modelling: Principles and Applications, Harlow 2001, Prentice Hall.

[6] D. Frenkel, B. Smit, Understanding Molecular Simulation: From Algorithms to Applications, San Diego 2002, Academic Press.

[7] J. Gasteiger, T. Engels, Chemoinformatics, Weinheim 2003, Wiley-VCH.

[8] A. R. Leach, V. J. Gillet, An Introduction to Chemoinformatics, Dordrecht 2007, Springer.

[9] G. Schneider, K.-H. Baringhaus, Molecular Design: Concepts and Applications, Weinheim 2008, Wiley-VCH.

[10] Examples of proprietary commercial applications are Statistica (http://www.statsoft.de), Origin (http://www.originlab.com) or SigmaPlot and TableCurve2D (http://www.sigmaplot.com).

[11] Examples of open-source frameworks are KNIME (http://www.knime.org) or Taverna (http://www.taverna.org.uk), Pipeline Pilot (http://accelrys.com) is a commercial implementation.

[12] Examples of open-source libraries are WEKA (http://www.cs.waikato.ac.nz/ml/weka) or the R Project (http://www.r-project.org).

[13] Examples are Maple (http://www.maplesoft.com), MATLAB (http://www.mathworks.com) or Mathematica (http://www.wolfram.com).

[14] Wolfram Mathematica, version 7 or higher (http://www.wolfram.com). Mathematica is licensed at most universities, research sites and major companies.

[15] *CIP*, version 1.2 for Mathematica 7 or higher, is available for download as a compressed ZIP file as supplementary material of this publication or at http://www.gnwi.de. This website also contains several tutorials with *CIP*-based data analysis workflows and a *CIP* user forum for discussion.

[16] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes: The Art of Scientific Computing, Cambridge 2007, Cambridge University Press, 3rd Edition.

[17] See online documentation at http://www.wolfram.com. The NonlinearModelFit command of Mathematica uses the Levenberg-Marquardt method for iterative unconstrained local minimization by default. See [2], [3] and [16] for algorithmic details.

[18] C. H. Reinsch, Smoothing by Spline Functions, Numer. Math. 10, 177-183, 1967. C. H. Reinsch, Smoothing by Spline Functions II, Numer. Math. 16, 451-454, 1971.

[19] L. Kaufman and P. J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, New York 1990, John Wiley & Sons.

[20] P. J. Rousseeuw, Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis, J. Comput. Appl. Math. 20, 53-65, 1987.

[21] *CIP* uses FindClusters with a method specification for partitioning around medoids [19]. If the number of resulting clusters k is not defined in advance the silhouette test is chosen to obtain the best k value [20].

[22] G. A. Carpenter, S. Grossberg, D. B. Rosen, ART 2-A: An Adaptive Resonance Algorithm for Rapid Category Learning and Recognition, Neural Networks 4, 493-504, 1991.

[23] D.Wienke, Y. Xie, P. K. Hopke, An adaptive resonance theory based artificial neural network (ART -2a) for rapid identification of airborne particle shapes from their scanning electron microscopy images, Chemometrics and Intelligent Laboratory Systems 25, 367-387, 1994.

[24] A. L. Edwards, An Introduction to Linear Regression and Correlation, San Francisco, CA 1976, W. H. Freeman.

[25] A. L. Edwards, Multiple Regression and the Analysis of Variance and Covariance, San Francisco, CA 1979, W. H. Freeman.

[26] S. Chatterjee, A. Hadi, B. Price, Regression Analysis by Example, New York 2000, John Wiley & Sons, 3rd Edition. Chapter 3: Multiple Linear Regression, pages 51-84.

[27] The Fit command performs least squares fits with linear combinations of functions [16, 24-26].

[28] J. A. Hertz, A. S. Krogh, R. G. Palmer, Introduction To The Theory Of Neural Computation, Redwood City, CA 1991, Addison-Wesley.

[29] R. Rojas, Neural Networks: A Systematic Introduction, Berlin 1996, Springer.

[30] The Polak-Ribiere variant of the Conjugate Gradient method [16].

[31] K. Price, R. Storn, Differential Evolution: Numerical Optimization Made Easy, Dr. Dobb's J. 264, 18-24, 1997.

[32] R. Storn, K. Price, Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, J. Global Optimization 11, 341-359, 1997.

[33] K. Price, An Introduction to Differential Evolution, 77-106, in: D. Corne, M. Dorigo, F.Glover (editors), New Ideas in Optimization, London 1999, McGraw-Hill.

[34] K. Price K., R. Storn, J. Lampinen, Differential Evolution - A Practical Approach to Global Optimization, Berlin 2005, Springer.

[35] J. A. Freeman, Simulating Neural Networks with Mathematica, Boston 1993, Addison-Wesley Longman Publishing Co.

[36] A. Forsgren, P. E. Gill, M. H. Wright, Interior Methods for Nonlinear Optimization, SIAM Rev. 44 (4), 525-597, 2002.

[37] S. Mehrotra, On the Implementation of a Primal-Dual Interior Point Method, SIAM J. Optimization 2, 575-601, 1992.

[38] The implementation is based on: B. Palancz, L. Volgyesi, Gy. Popper, Support Vector Regression via Mathematica, Periodica Polytechnica Civ. Eng 49 (1), 59-84, 2005.

[39] The Mathematica notebook file with the complete workflow is available as supplementary material of this publication and will also be made available after publication at website http://www.gnwi.de.

[40] Wisconsin Diagnostic Breast Cancer (WDBC) data set. Taken at 2011/01/30 from the UCI (University of California at Irvine) machine learning repository at http://archive.ics.uci.edu/ml. Repository citation: A. Frank, A. Asuncion (2010), UCI Machine Learning Repository (http://archive.ics.uci.edu/ml), Irvine, CA: University of California, School of Information and Computer Science. First citation in medical literature: W.H. Wolberg,W.N. Street, and O.L.Mangasarian, Machine learning techniques to diagnose breast cancer from fine-needle aspirates, Cancer Letters 77 (1994), 163-171.

[41] Details are discussed in: A. Zielesny, From Curve Fitting to Machine Learning: An illustrative Guide to scientific Data Analysis and Computational Intelligence, Springer: Intelligent Systems Reference Library, Volume 18, Berlin, 2011 (http://dx.doi.org/10.1007/978-3-642-21280-2), chapters 3 and 4. CIP, version 1.0, was used for all examples and applications of this textbook.

Figure 1. Complete workflow document with expanded and collapsed sections.

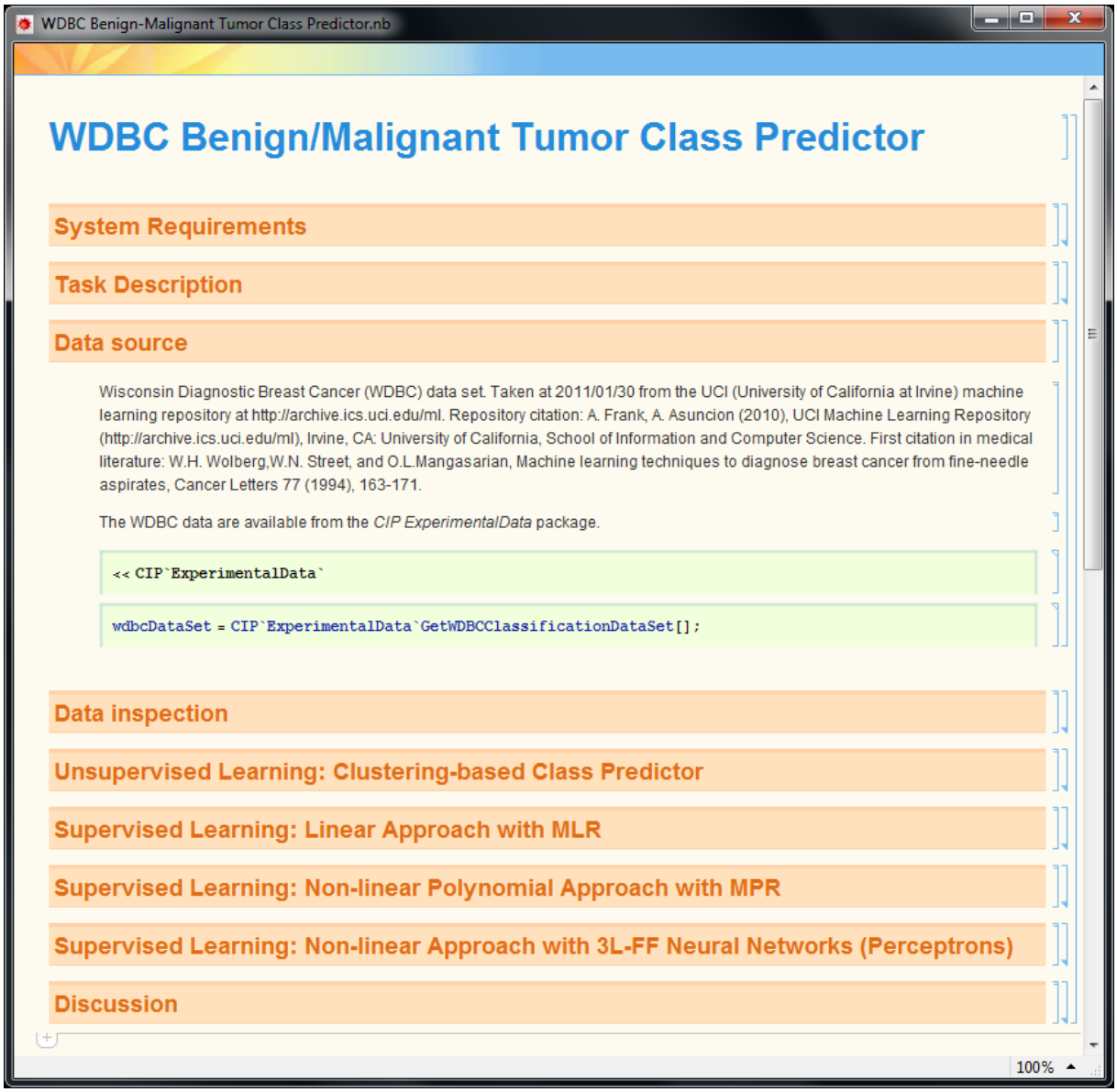


Figure 2. Expanded section "Data inspection".

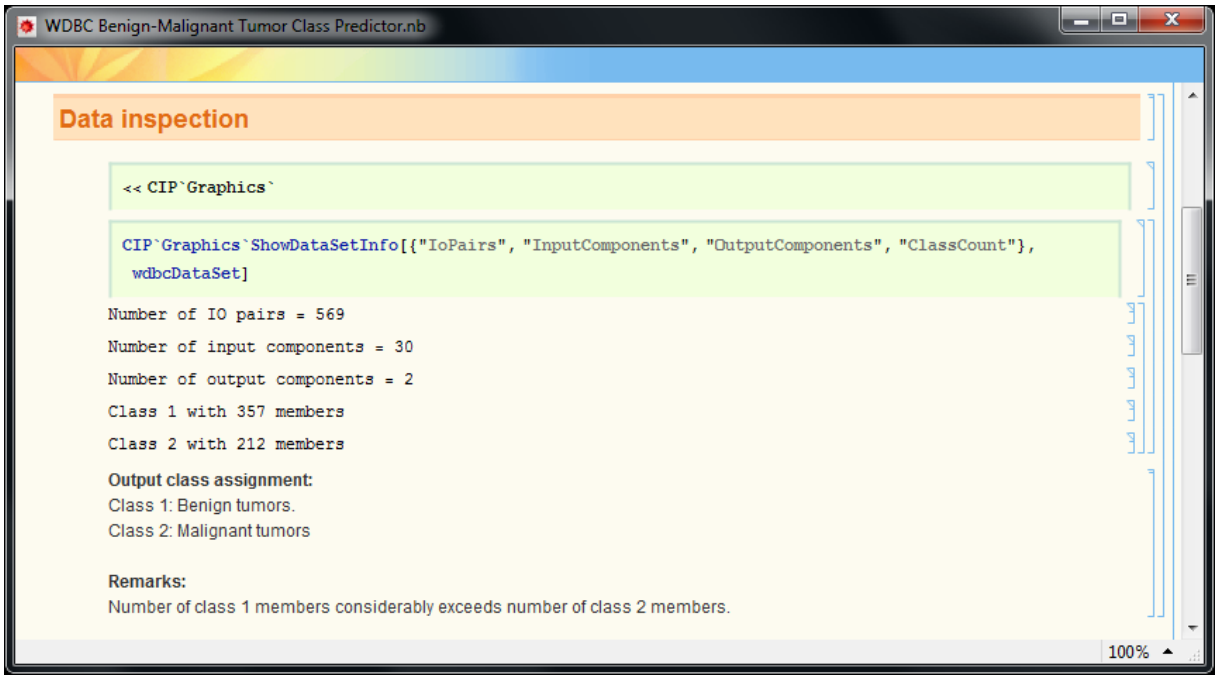


Figure 3. Expanded section "Unsupervised learning …".

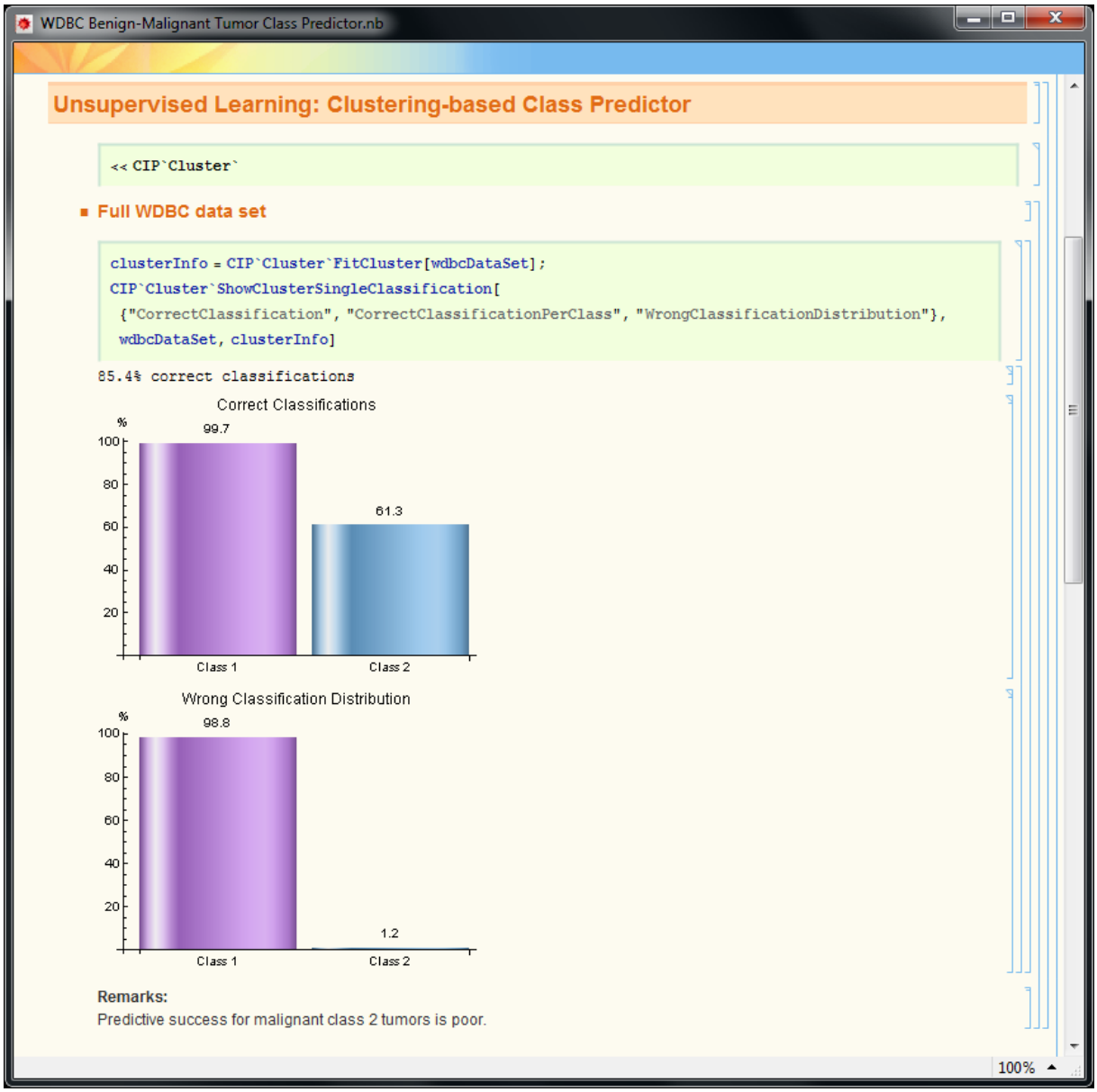


Figure 4. Partly expanded section "Supervised Learning … MLR".

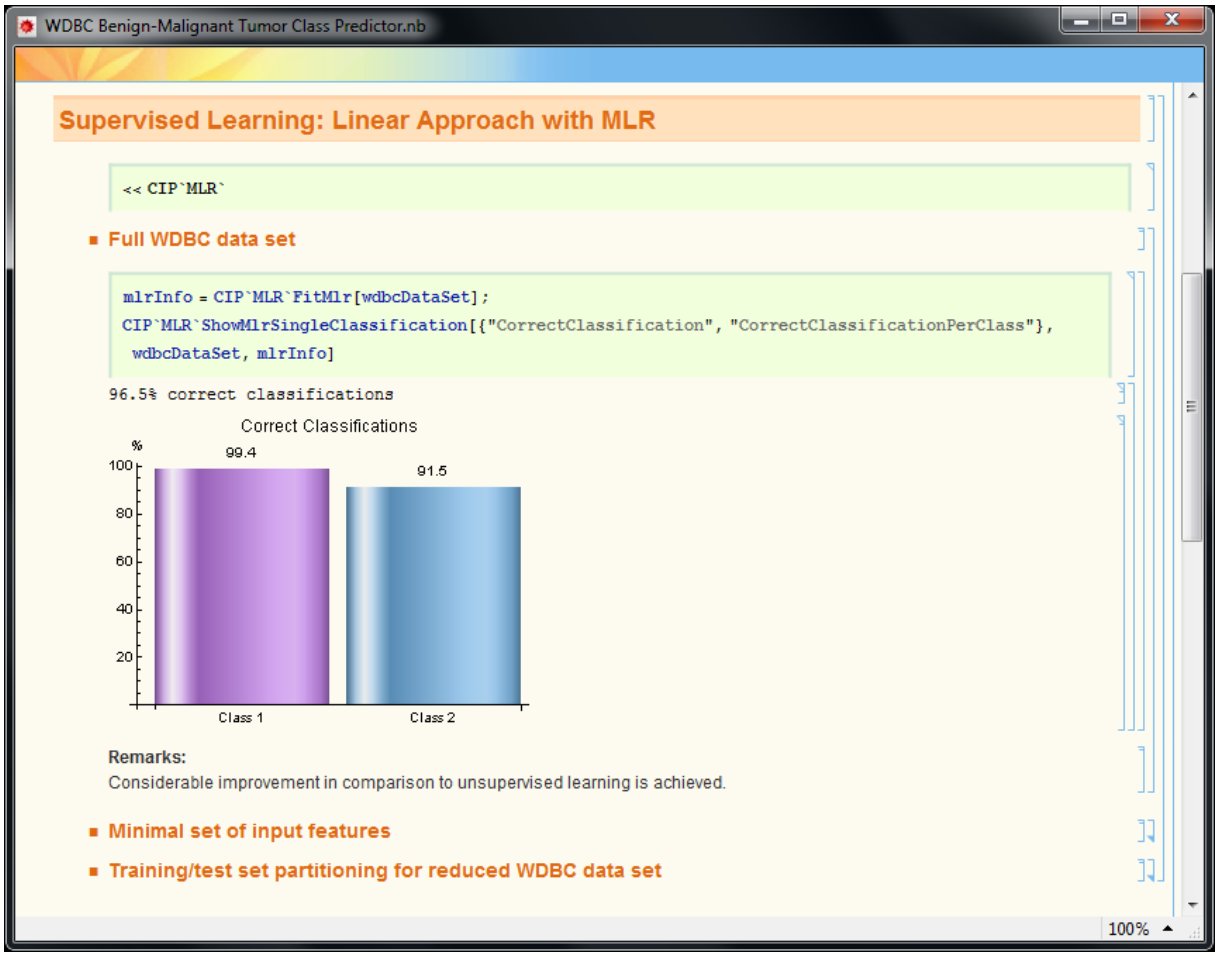


Figure 5. Expanded MLR subsection "Minimal set of input features".

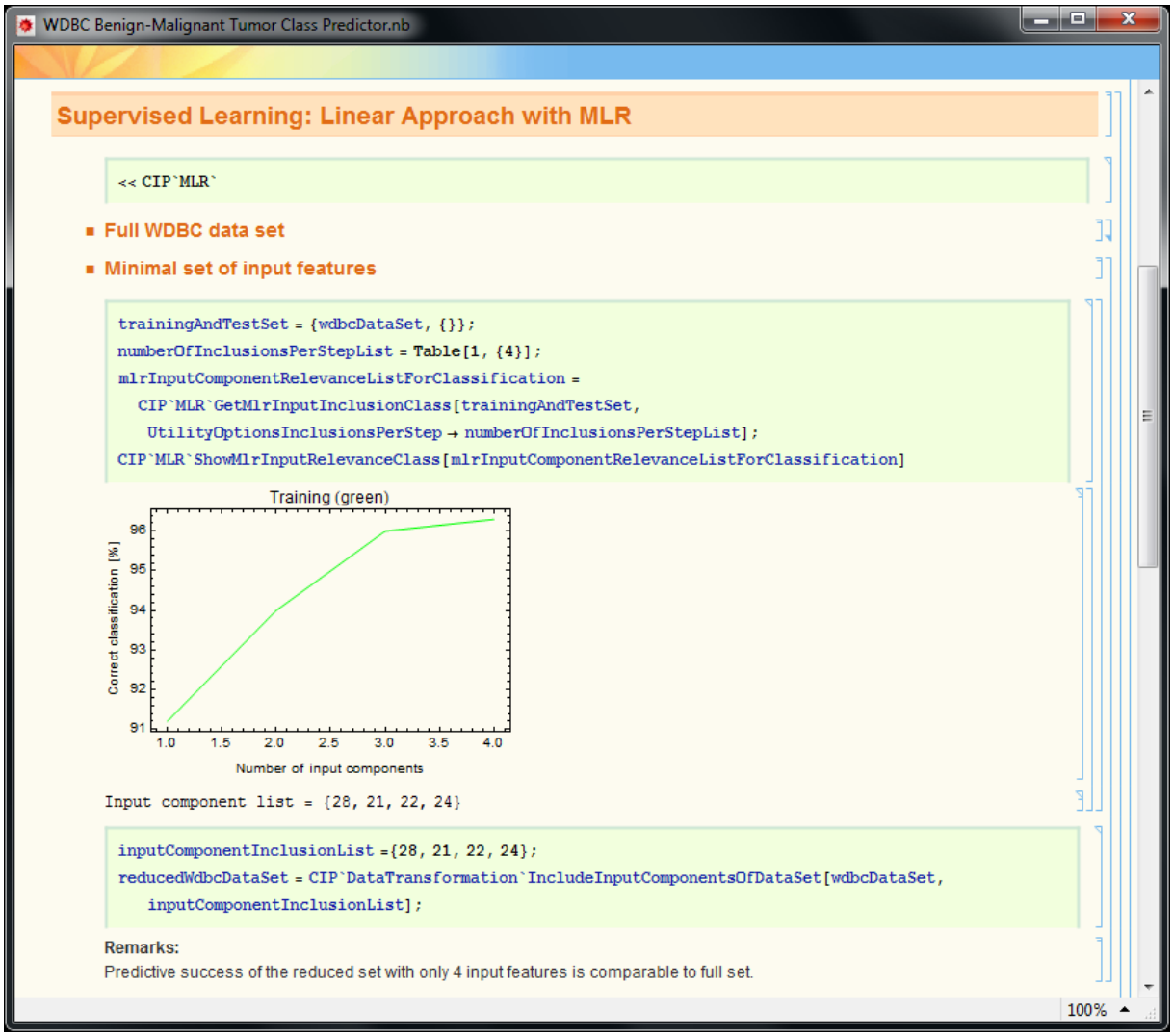


Figure 6. Expanded MLR subsection "Training/test set partitioning …".

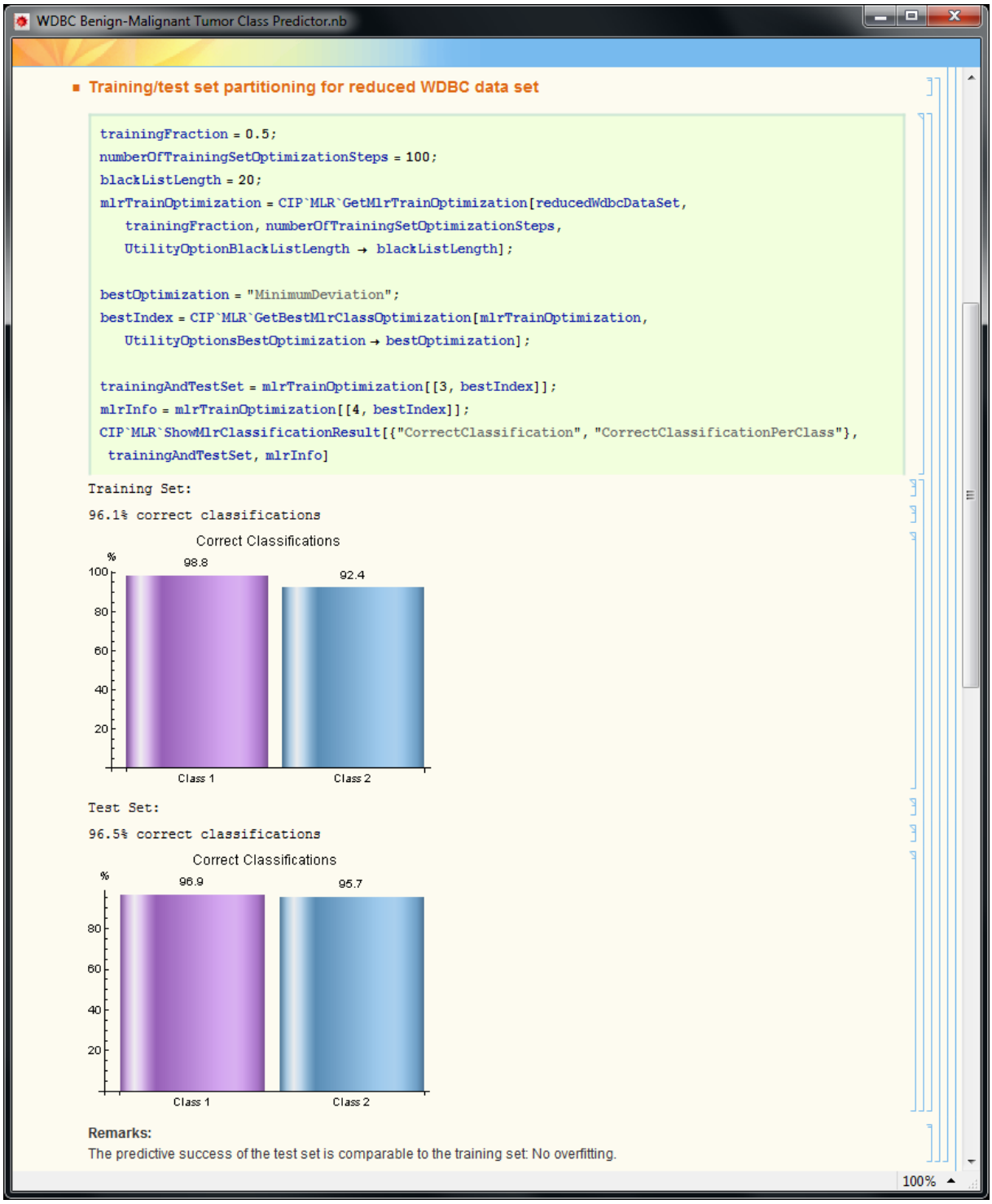


Figure 7. Overfitting with non-linear parabolic MPR approach

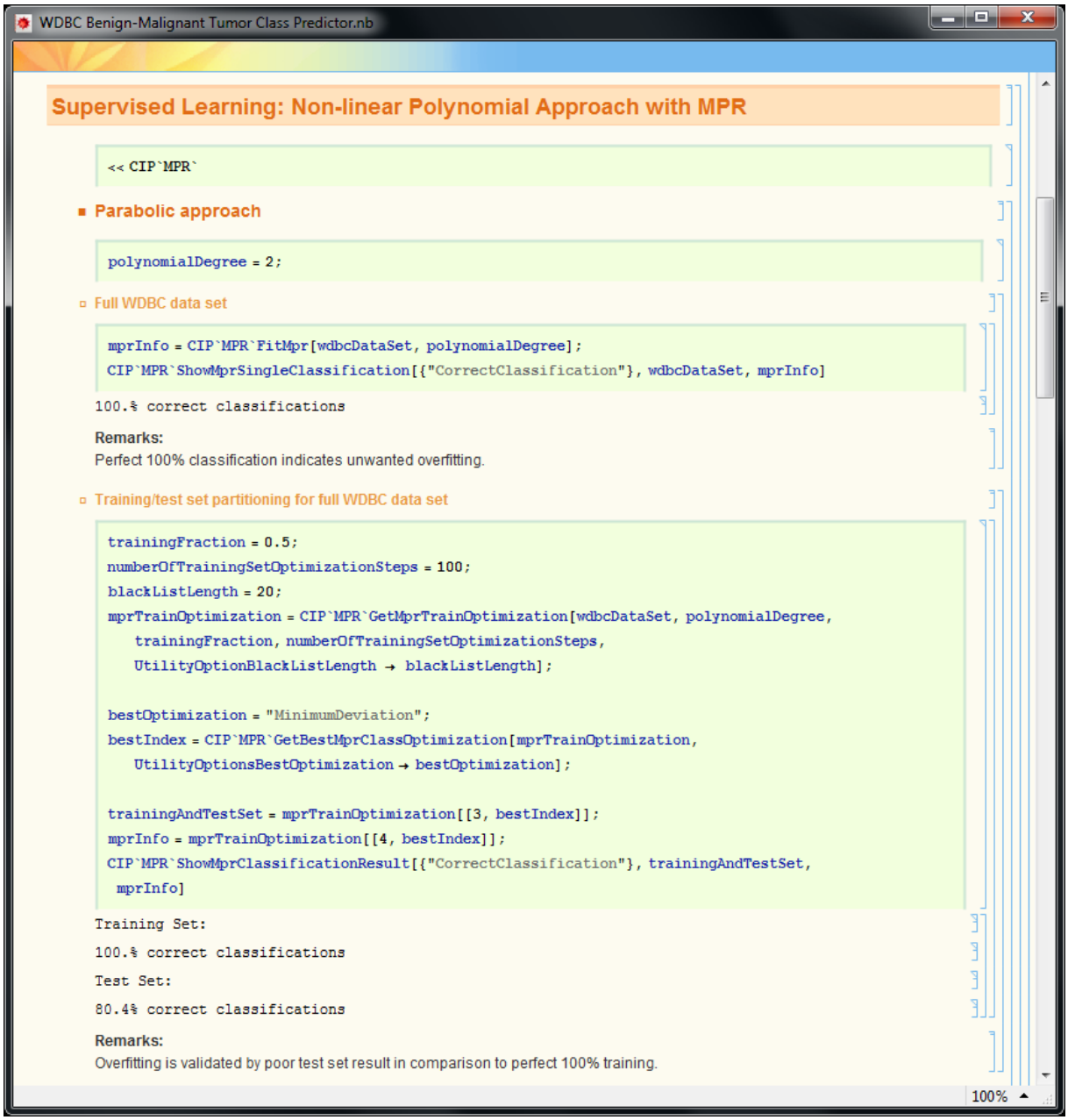


Figure 8. Expanded MPR subsection "Cubic approach".

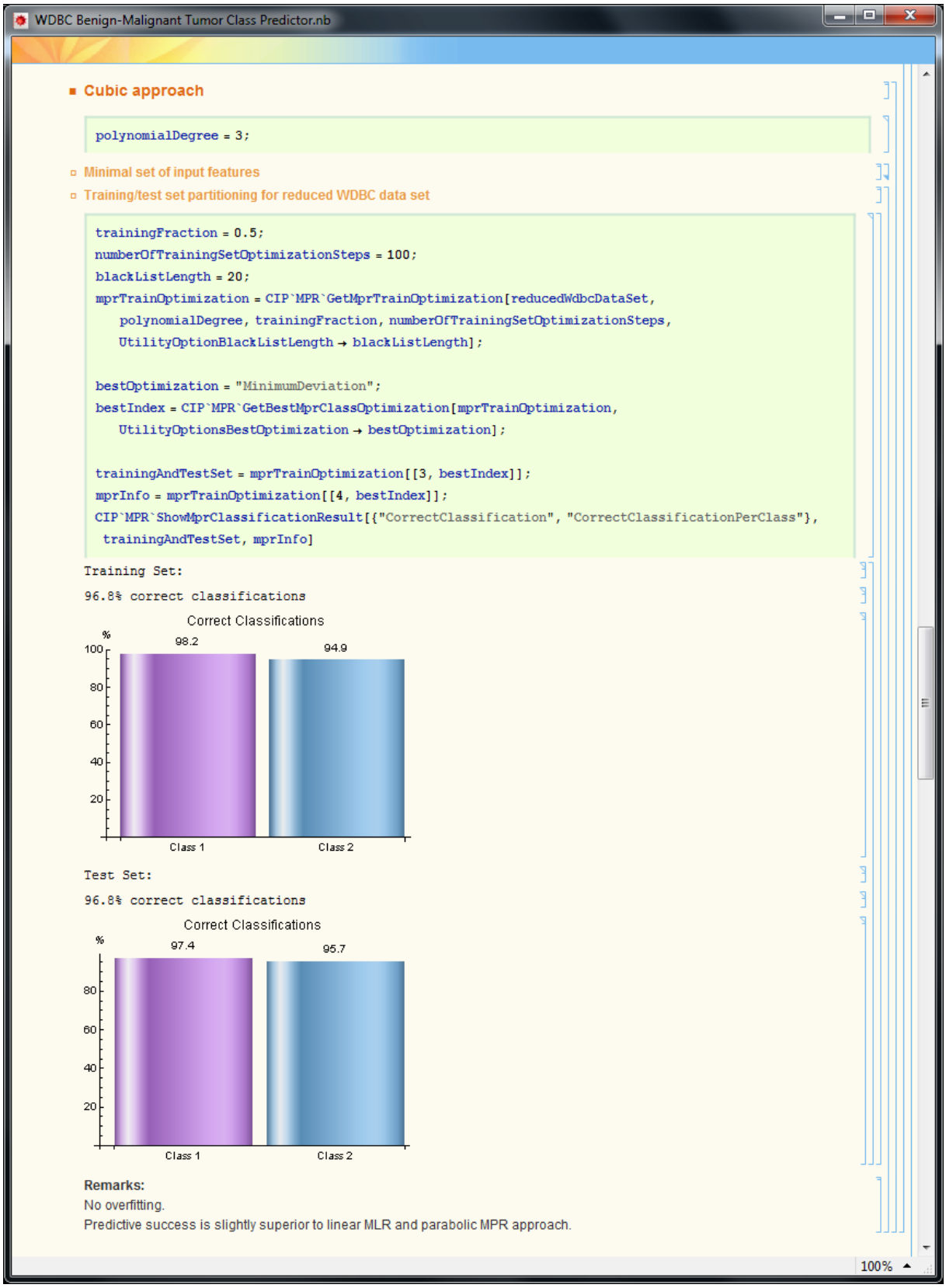


Figure 9. Expanded Perceptron subsection "3 hidden neurons …".

WDBC Benign-Malignant Tumor Class Predictor.nb

3 hidden neurons (3-HN)

```
numberOfHiddenNeurons = 3;
```

Minimal set of input features

Training/test set partitioning for reduced WDBC data set

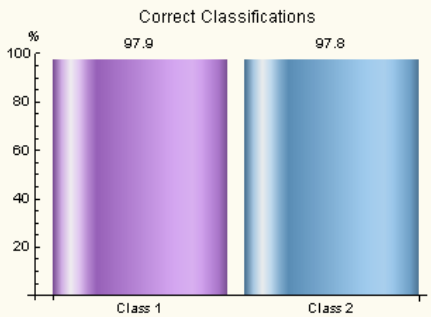
```
trainingFraction = 0.5;
numberOfTrainingSetOptimizationSteps = 100;
blackListLength = 20;
perceptronTrainOptimization = CIP`Perceptron`GetPerceptronTrainOptimization[
   reducedWdbcDataSet, numberOfHiddenNeurons, trainingFraction,
   numberOfTrainingSetOptimizationSteps, UtilityOptionBlackListLength → blackListLength];

bestOptimization = "MinimumDeviation";
bestIndex = CIP`Perceptron`GetBestPerceptronClassOptimization[perceptronTrainOptimization,
   UtilityOptionsBestOptimization → bestOptimization];

trainingAndTestSet = perceptronTrainOptimization[[3, bestIndex]];
perceptronInfo = perceptronTrainOptimization[[4, bestIndex]];
CIP`Perceptron`ShowPerceptronClassificationResult[
 {"CorrectClassification", "CorrectClassificationPerClass"}, trainingAndTestSet,
 perceptronInfo]
```

```
Training Set:
97.9% correct classifications
```


```
Test Set:
97.9% correct classifications
```



Remarks:
No overfitting.
Predictive success is slightly superior to cubic MPR approach.

100%