**Building a Simple classifier for well logs in IP**

Summary:

In this report, the steps on how to build the classifier are shown. First, the curves can be saves as txt files and loaded to visual studio. Accord machine learning and math libraries are used, and three different classification algorithms(Naïve Bayes, Decision tree and logistic regression) are tested. The decision tree algorithm is used to develop the app in IP(interactive petrophysics), where cross validation, grid search and metrics for performance are all used to build the model. The APP results are shown to demonstrate the success of the implementation.

1. Save data from Interactive Petrophysics

LR’s IP has some built-in APIs to call to do curves manipulation. It also has some APIs generating pop up windows messages) from PGL.IP.API, PGL.IP.SERVICE etc. For example, messageBoard will pop up a window while message can display some values (curve values) for primitive interactive variable watch for debugging purpose.

The following example shows how to save some curves to local txt files, and they can be used for model building with curves from other wells.

using System;

using System.Collections.Generic;

using System.Data;

using System.Diagnostics;

using System.Windows.Forms;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Numerics;

using PGL.IP.API;

using PGL.IP.Services;

//using Accord;

//using Accord.Math;

//using Accord.IO;

//using Accord.MachineLearning.Bayes;

//using Accord.Controls;

using PGL.IP.UserProgDotNetInterface;

public partial class IPLink

{

public void UserCode()

{

List<float> gr = new List<float>();

List<float> rdeep = new List<float>();

List<float> depth = new List<float>();

List<int> rhobBadFlag = new List<int>();

// var api=new IntPetroAPI(); //get api instance

//IMessageBoard messageBoard = api.GetService<IMessageBoard>();

StringBuilder message = new StringBuilder();

var data = new DataTable();

data.Columns.Add("depth","gr", "rdeep","bhflag");

int index;

//

// Loop through the data one level at a time

// TopDepth and BottomDepth are the index equivalent depths entered on the run window.

//

for (index = TopDepth; index <= BottomDepth; index+=50)

{

// Enter user code here

if ((GR(index) != -999) && (RDEEP(index) != -999))

{

gr.Add(GR(index));

rdeep.Add(RDEEP(index));

depth.Add(Depth(index));

int bhflag=1;

if ( (GR(index)< 150) && (RDEEP(index)<1500))

{

bhflag=0;

}

rhobBadFlag.Add(bhflag);

data.Rows.Add(Depth(index),GR(index),RDEEP(index), bhflag);

//message.AppendFormat("Depth: {0}\t Gamma Value: {1:0.00}\t resistivity Value:{2:0.00} \n", Depth(index), GR(index), RDEEP(index));

message.AppendFormat("Depth: {0}\t Gamma Value: {1:0.00}\t resistivity Value:{2:0.00}\t flag:{3:0.00} \n", Depth(index), GR(index), RDEEP(index),bhflag);

}

}

//MessageBox.Show(message.ToString());

File.WriteAllLines("mygr.txt", gr.Select(x => x.ToString()).ToArray());

File.WriteAllLines("myrdeep.txt", rdeep.Select(x => x.ToString()).ToArray());

File.WriteAllLines("mydepth.txt", depth.Select(x => x.ToString()).ToArray());

File.WriteAllLines("mybadflag.txt", rhobBadFlag.Select(x => x.ToString()).ToArray());

//double[][] inputs = data.ToJagged<double>("gr", "rdeep");

//int[] outputs = data.Columns["bhflag"].ToArray<int>();

}

}

To incorporate wells information or read from multiple wells, one needs to use IP’s API, for example, in the following code, an instance is initiated, and the dataset, well and curveset information are obtained and the curves in the specific well are loaded and then can be saves as in the above example:

var api=new IntPetroAPI(); //get api instance

IMessageBoard messageBoard = api.GetService<IMessageBoard>();

IDatabaseFactory dbFactory = api.GetService<IDatabaseFactory>();

IDatabase database = dbFactory.GetConnectedDatabase();

IWell well= database.ActiveWell;

messageBoard.Add(MessageType.Information,"Active Well:"+well.DisplayLabel);

//ICurveSet curveSet = well.NewCurveSet("UserAppDemo"); // create a new curve set

//ICurveSet curveSet2 = well.FindCurveSet();

//ICurve NewCurve(string newgr);

// getting a curveset

ICurve gamma = well.FindCurve("GR"); // use quote to identify curve name

ICurve res = well.FindCurve("RDEEP"); // use quote to identify curve name

ICurve nphil = well.FindCurve("N\_PHI"); // use quote to identify curve name

ICurve dt = well.FindCurve("DT"); // use quote to identify curve name

//messageBoard.Add(MessageType.Information,gamma.FullName); // get curve

//messageBoard.Add(MessageType.Information,res.FullName); // get curve

//messageBoard.Add(MessageType.Information,nphil.FullName); // get curve

//messageBoard.Add(MessageType.Information,dt.FullName); // get curve

//MessageBox.Show(gamma.FullName, "get wells");

ICurveSet gammaCurveSet = gamma.CurveSet;

int displayCount = 5;

StringBuilder message = new StringBuilder();

//Read the values using the gamma ray sample rate

int topDepthIndex = gammaCurveSet.DepthCurve.GetIndex(gammaCurveSet.DepthCurve.TopDepth);

int BottomDepthIndex = gammaCurveSet.DepthCurve.GetIndex(gammaCurveSet.DepthCurve.BottomDepth);

for (int depthIndex = topDepthIndex; depthIndex < topDepthIndex + displayCount; depthIndex++)

{

double depth = gammaCurveSet.DepthCurve.Value(depthIndex, gammaCurveSet.ID);

double grValue = gamma.Value(depthIndex, gammaCurveSet.ID);

double resValue = res.Value(depthIndex, gammaCurveSet.ID);

double nphilValue = nphil.Value(depthIndex, gammaCurveSet.ID);

double dtValue = dt.Value(depthIndex, gammaCurveSet.ID);

message.AppendFormat("Depth: {0}\t Gamma Value: {1:0.00}\t resistivity Value:{2:0.00} dt Value: {1:0.00}\t nphil Value:{2:0.00}\n", depth, grValue, resValue,dtValue,nphilValue);

}

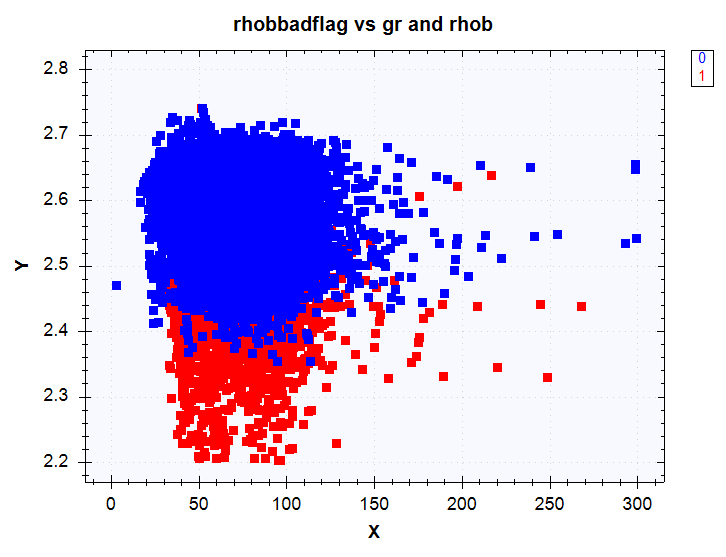
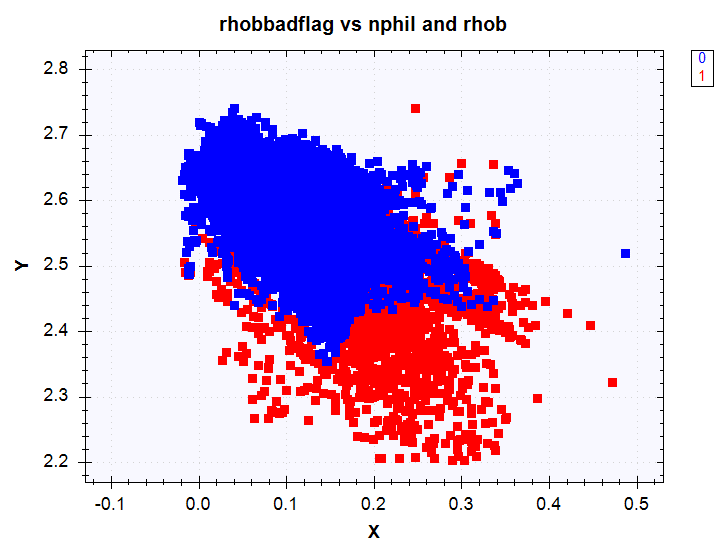
1. Model Building

After saving the curves from IP to local directory, the classification model building can be done locally without interacting with IP, especially if there are many curves from different wells are used as training data.

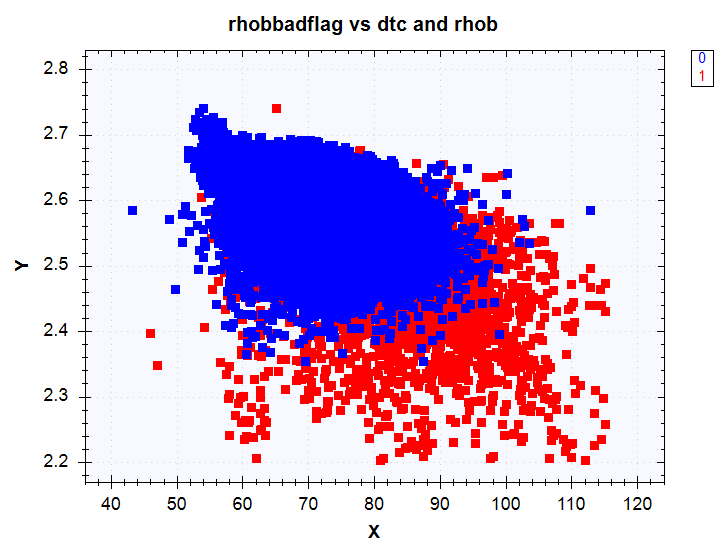
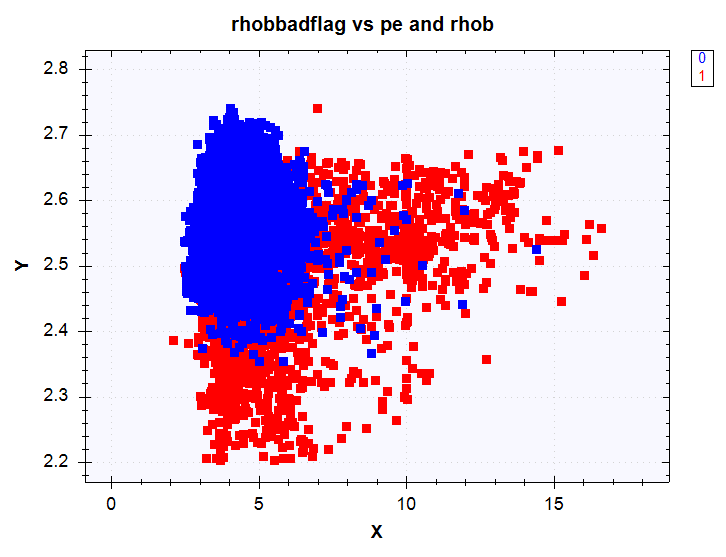
In the following, three classification methods are tested to see the possibility of building predictive model for labeled data. Since no sources of labeled data is available at hand, the one with badhole flags (which has 1 or 0) is used as a simple demonstration of the application. There might not be any intrinsic link or physics-based for the model (this is purely from an example data)

1. Scatter plots

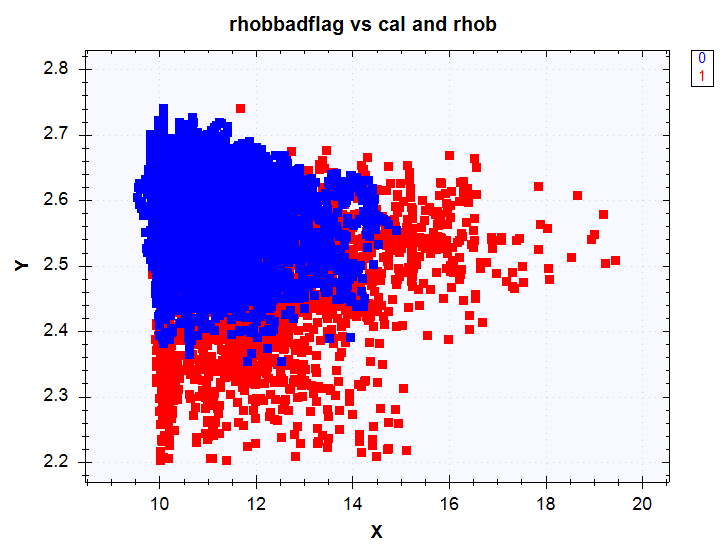
First, Accord’s API scatterbox is used to plot the relations among the variables to see if there’s any possibilities of classification. All the data removed the N/A values (-999) so this value is not shown in the analysis and plots.

Rhobbadflags, x axis is gr, y axis is rhob. Rhobbadflags, x axis is nphil, y axis is rhob.

Rhobbadflags, x axis is dtc, y axis is rhob Rhobbadflags, x axis is pe, y axis is rhob



Rhobbadflags, x axis is cal, y axis is rhob

Fig.1. rhobbadflag vs different combination of variables.

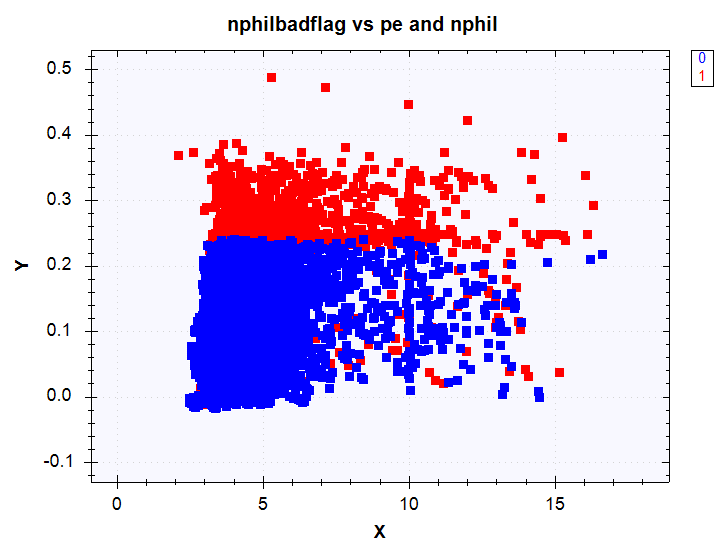
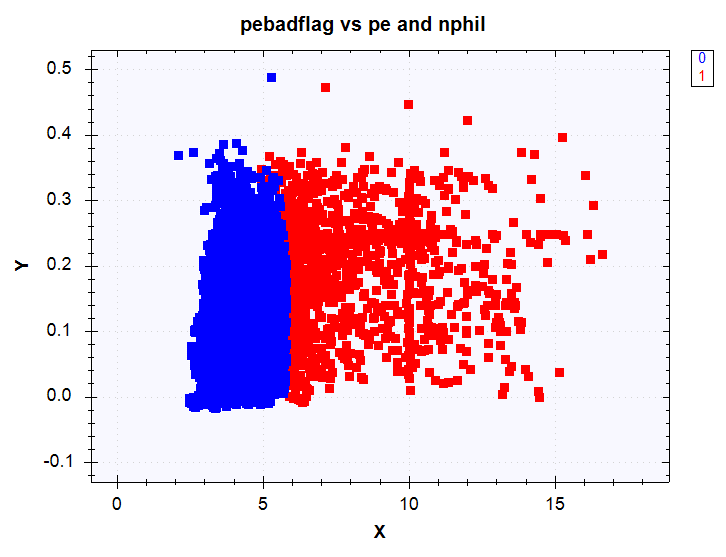
From the above plots, we can sees that a possible classification model can be built because

there seems a separation of “1” and “0” for the rhobflag for variables “rhbo” and “gr” or

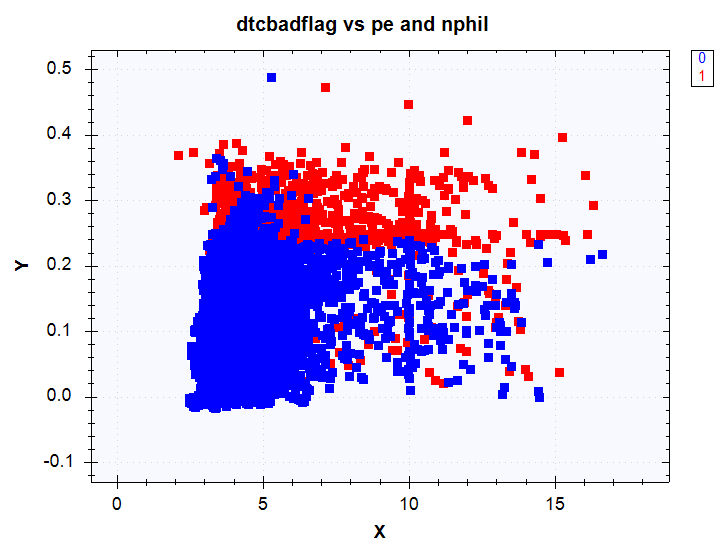
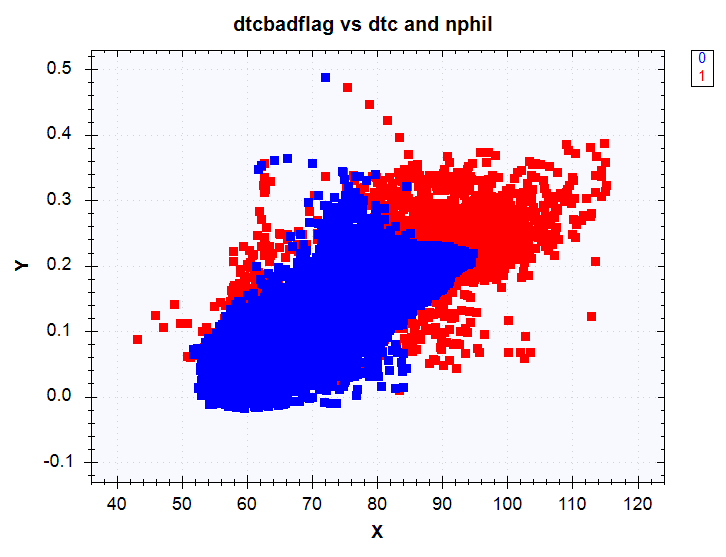
“nphil” or “dtc” or “pe” or “cal”, or maybe they all contribute to the classification (here it is just

an assumption, there is no base for this assumption). However, it is hard to see the high-

dimension relationship of all variables, so here only two variables are plotted as demonstration.

nphilbadflags, x axis is pe, y axis is nphil pebadflags, x axis is pe, y axis is nphil

dtclbadflags, x axis is pe, y axis is nphil dtclbadflags, x axis is dtc, y axis is nphil

Fig.2. Different badflag vs different combination of variables.

Fig.2. shows that other labelled curve(data) can also be studied. The labelled data is not generally a good example here because of the heavy overlapping. It is shown here only for demonstration.

1. Classification model building:

There are many different methods for classification. Here only three are picked for demonstration purpose, logistic regression, naïve bayes, decision tree are shown here. Other methods like SVM, neural network, random forest can be tested later to see which one provide the best accuracy.

For simple model building demonstration, two input variables are used (but it’s not limited to two), “gr” and “rhob”. The prediction label is “rhobbadflag”. To save time on model building, only 1/3 of the data is used, and 75% is used as training, and 25% is used for testing.

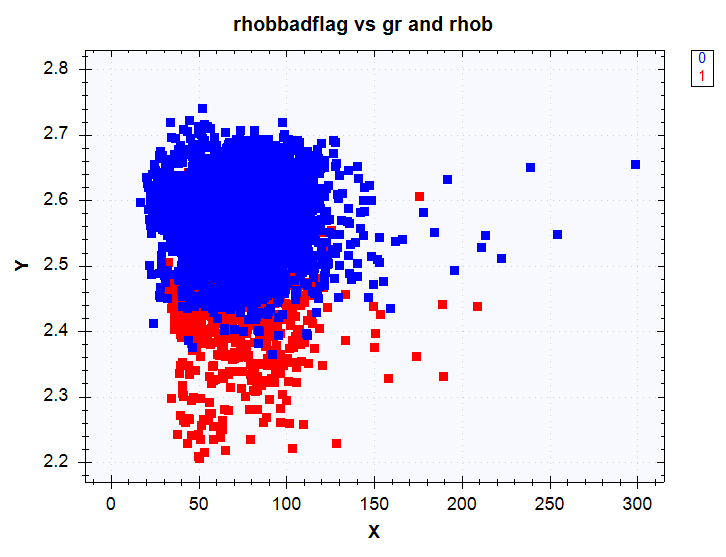


Fig.3. The dataset used for model building

1. Naïve Bayes

Naïve Bayes classifiers are a family of probabilistic classifiers based on Bayes’ theorem with strong independence assumptions between the features. Here, it is assumed that “gr” and “rhob” are the independent features. The predicted object is rhobbadflag, and it can either be “1” (bad hole) or “0” (not bad). In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of “1” and “0” objects, and often used to predict outcomes before they actually happen.

**Prior Probability of**1: number of 1 objects / total number of objects

**Prior Probability of**0: number of 0 objects / total number of objects

Then calculate likelihood: to measure this likelihood, we draw a circle around X which encompasses a number (to be chosen a priori) of points irrespective of their class labels. Then we calculate the number of points in the circle belonging to each class label.

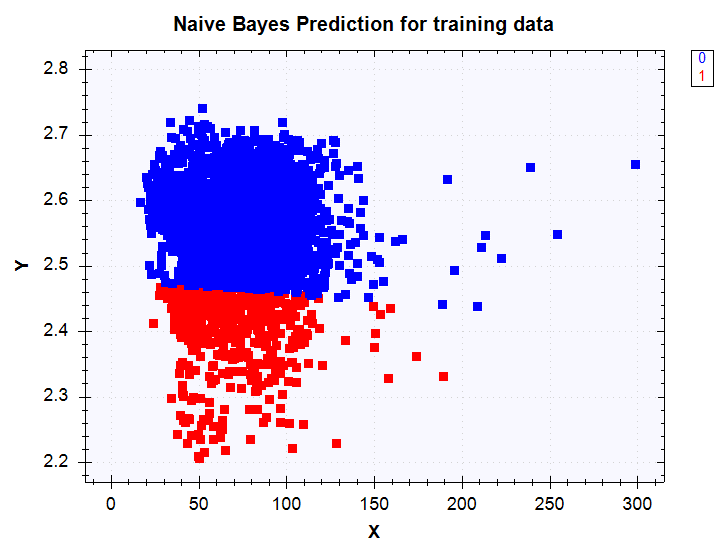
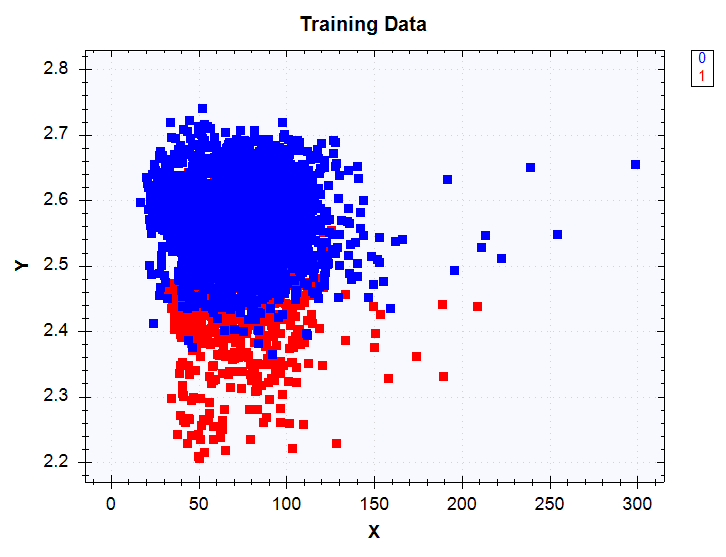
**Likelihood of**X given “1”: number of 1 in the vicinity of X / total number of 1 objects

**Likelihood of**X given “0”: number of 0 in the vicinity of X/ total number of 0 objects

In the Bayesian analysis, the final classification is produced by combining both sources of information, i.e., the prior and the likelihood, to form a posterior probability using the so-called Bayes' rule (named after Rev. Thomas Bayes 1702-1761).

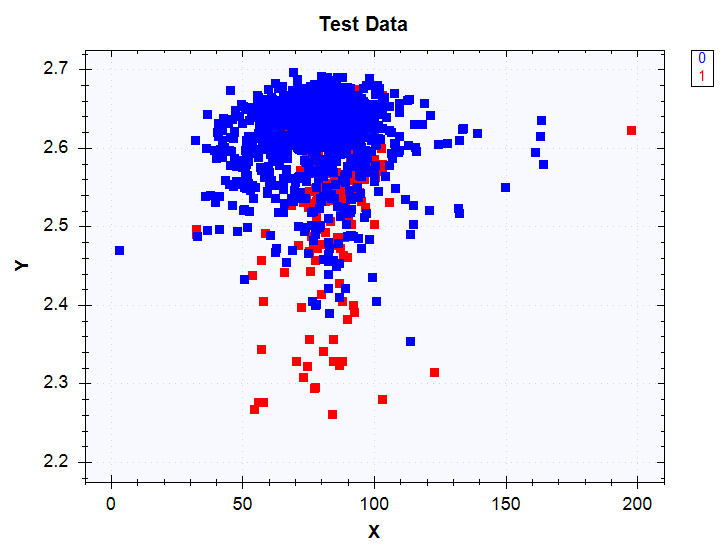
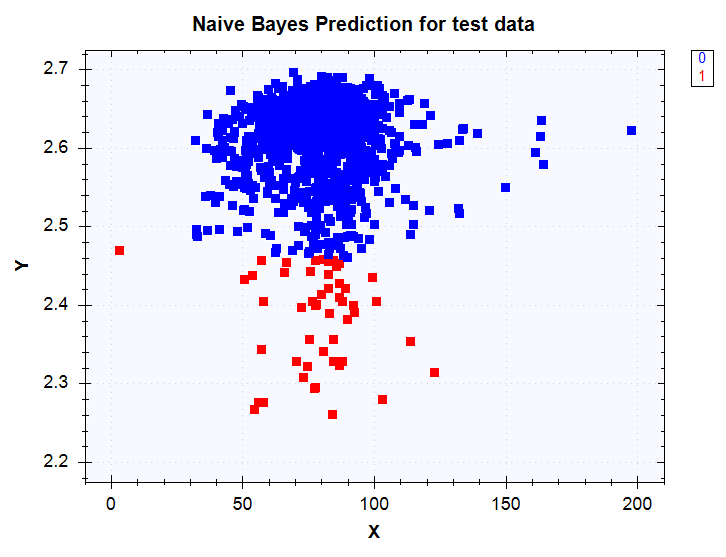
**Posterrior Probability of X being**1: Prior probability of 1 X Likelihood of X given 1

**Posterrior Probability of X being** 0: Prior probability of 0 X Likelihood of X given 0



Training data (original data) predicted data (for training data)

Fig. 4. Naïve Bayes for training data, classification demonstration.

Test data (original data) predicted data (for test data)

Fig. 5. Naïve Bayes for testing data, classification demonstration.

1. Decision Tree

Decision Trees give a direct and intuitive way for obtaining the classification of a new instance

from a set of simple rules. The trees can be drawn by hand based on prior knowledge, or

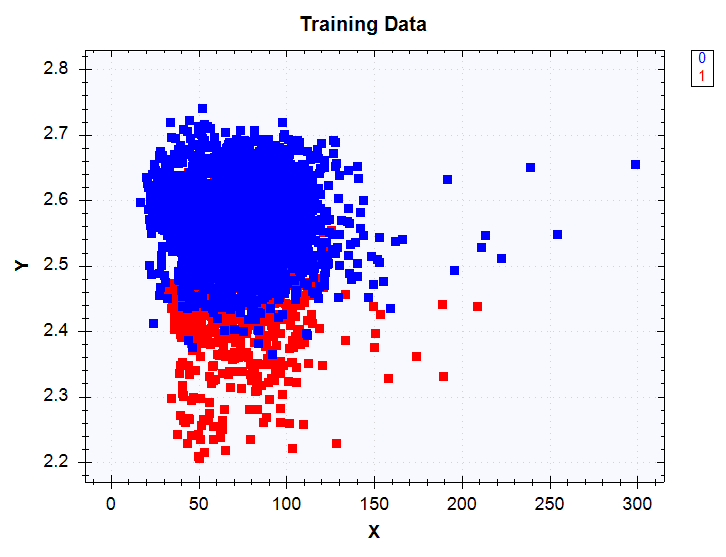
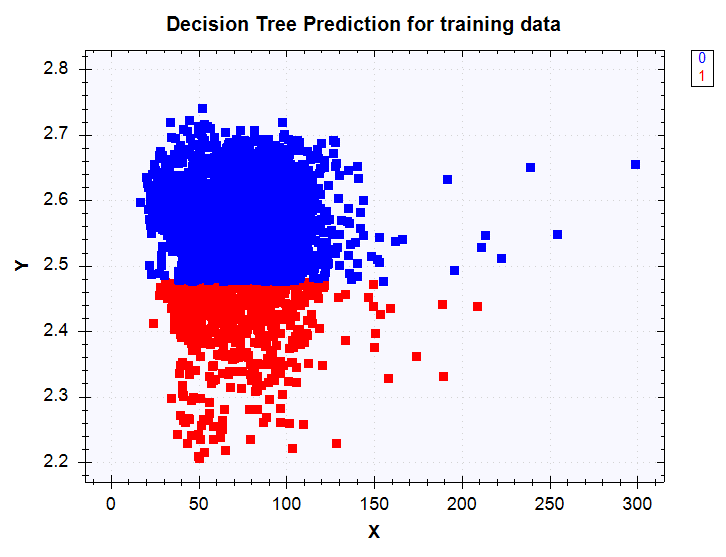
learned automatically through learning algorithm.

Accord machine learning decisiontrees class has ID3 and C45 algorithms, both perform local

optimum decisions in producing a most general tree. Such algorithms are based on the principle

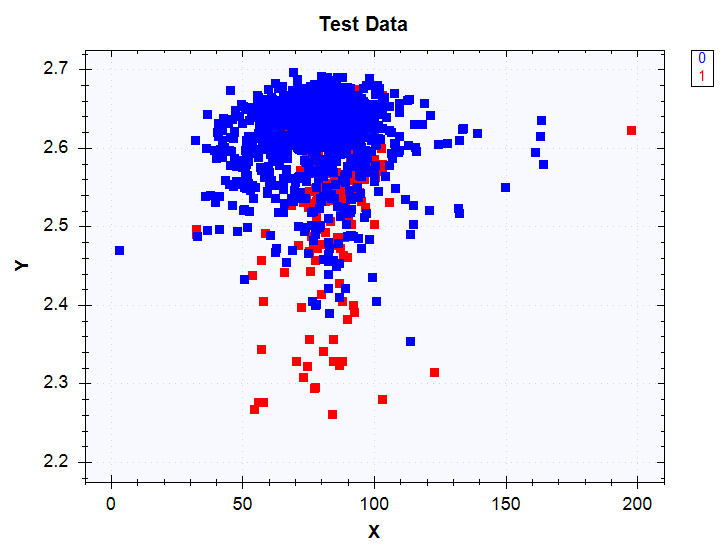
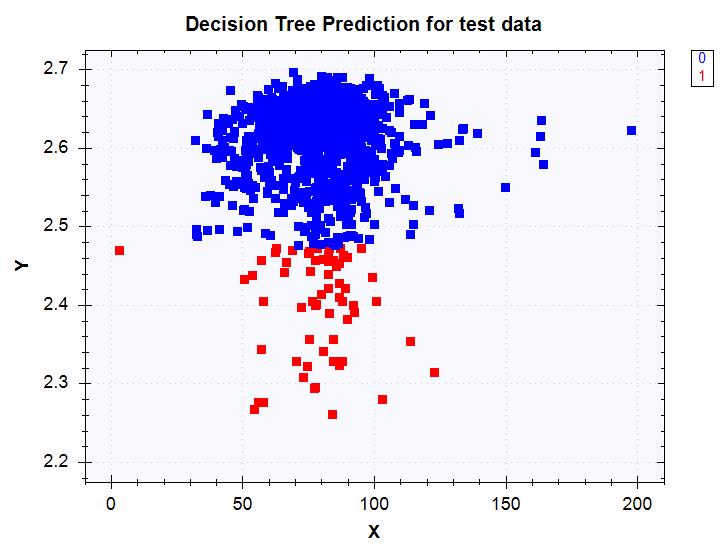
of the Occam’s Razor, favoring simpler trees retaining more generalization power. The

following results are generated from c45 algorithm.

Training data (original data) predicted data (for training data)

Fig. 6. Decision Tree for training data, classification demonstration.

Test data (original data) predicted data (for test data)

Fig. 7. Decision Tree for training data, classification demonstration.

1. Logistic Regression

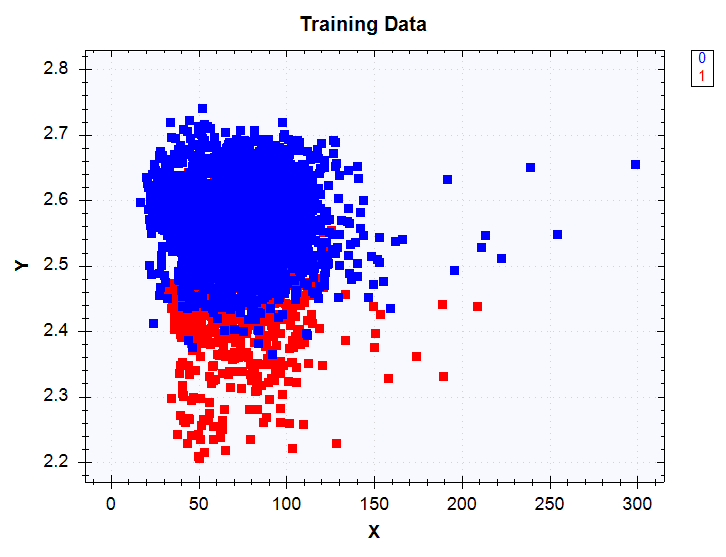
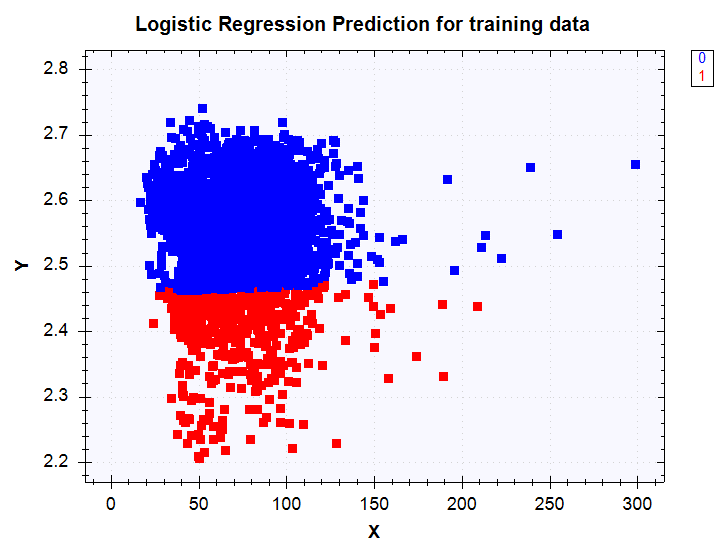
Logistic regression classifier is for binary classification problems (with two class values only).

So if the classes are more than two, this algorithm can no longer be used. Unlike linear

regression which outputs continuous number values, logistic regression transforms its output

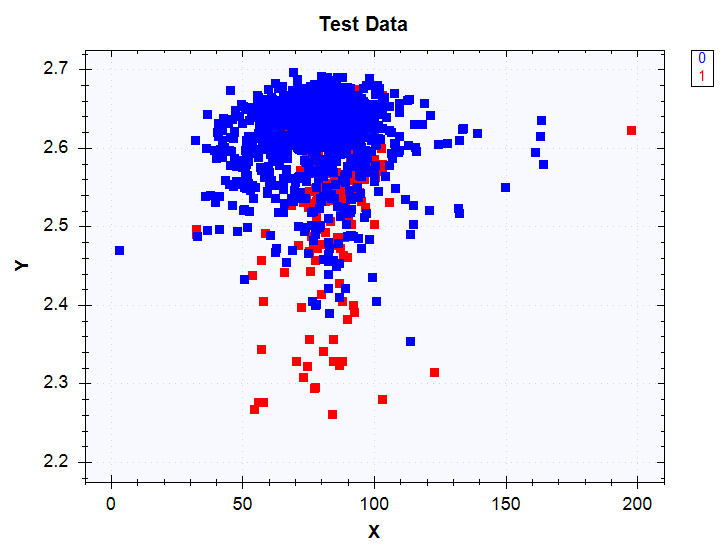
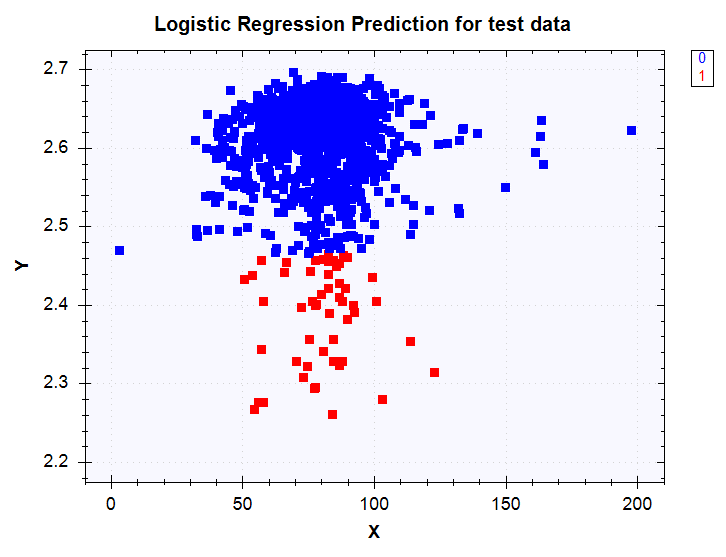
using the logistic sigmoid function to return a probability value which can then be mapped to

two or more discrete classes.

Training data (original data) predicted data (for training data)

Fig. 8. Logistic Regression for training data, classification demonstration.

Test data (original data) predicted data (for test data)

Fig. 9. Logistic regression for training data, classification demonstration.

From Fig. 4-9, we demonstrate three methods(Naïve Bayes, decision tree, logistic regression

classifier) to classify rhobbadflag through two feature gamma ray and resistivity curves. The ideal

classification data would be no overlapping data. Here it is only for demo. We see that the models

did separate the “1” and “0” from training data, and applied to testing data and separate the two

classes.

1. Model Evaluation using Confusion Matrix

There are many metrics for classification to measure the performance of the predictor, mainly derived from confusion matrix. Accord.Statistics.Analysis has GeneralConfusionMatrix class to evaluate the classification. The basic diagram of the confusion matrix is shown in Fig. 10. Some of the commonly used ones include accuracy, precision, recall, FScore etc.

So here are the definition for the metrics:

Accurary: (TP+TN)/total, The overall agreement is the sum of the diagonal elements of the contingency table divided by the number of samples.

Error rate: (FP+FN)/total

ChangeAgreement: The chance agreement tells how many samples were correctly classified by chance alone.

GeometryAgreement: The geometric agreement is the geometric mean of the diagonal elements of the confusion matrix

Pearson: Pearson's C measures the degree of association between the two variables. However, C suffers from the disadvantage that it does not reach a maximum of 1 or the minimum of -1; the highest it can reach in a 2 x 2 table is .707; the maximum it can reach in a 4 × 4 table is 0.870. It can reach values closer to 1 in contingency tables with more categories. It should, therefore, not be used to compare associations among tables with different numbers of categories

Kappa: This is essentially a measure of how well the classifier performed as compared to how well it would have performed simply by chance. In other words, a model will have a high Kappa score if there is a big difference between the accuracy and the null error rate.

Tau: Tau-b statistic, unlike tau-a, makes adjustments for ties and is suitable for square tables. Values of tau-b range from −1 (100% negative association, or perfect inversion) to +1 (100% positive association, or perfect agreement). A value of zero indicates the absence of association.

ChiSquare: is used to determine whether two variables are independent of one another. If the result is above a given critical threshold value then we can say that there is a relationship between the variables, otherwise there is no relation. Given a classification rule we can determine whether the rule is surprising (i.e. unexpected) or not by determining whether there exists some special relationship between the attributes and the classifier, or that the rule is simply one that we might expect assuming a normal (*chi-squared*) distribution.

The computed metrics for three different methods are compared in Table 1. It is observed that there is not too much difference in terms of performance among the three.

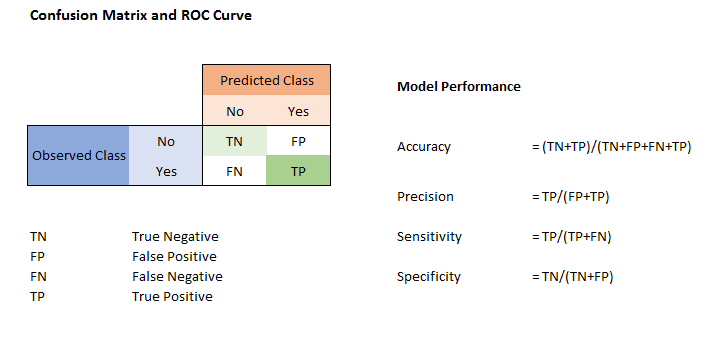


Fig. 10. A confusion matrix definition and some metrics.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Naïve Bayes | Decision Tree | Logistic Regression |
| Accuracy | 0.87714285714285711 | 0.87571428571428567 | 0.87857142857142856 |
| Error | 0.12285714285714289 | 0.12428571428571433 | 0.12142857142857144 |
| ChangeAgreement | 0.84224489795918367 | 0.83592653061224487 | 0.84119183673469389 |
| GeometryAgreement | 192.63177308014375 | 206.97825972792404 | 198.74858490062263 |
| Pearson | 0.26810248177031065 | 0.27515650217371729 | 0.28074444182486075 |
| Kappa | 0.22121604139715378 | 0.24249962684710669 | 0.23537575819882789 |
| Tau | 0.29693430656934294 | 0.33360324772670058 | 0.3096006126569352 |
| ChiSquare | 108.42391304347827 | 114.67793353660065 | 119.78561389101158 |

Table 1. Comparison of Metrics from confusion matrix for three different methods

1. Cross validation and grid search for tuning

Cross-validation is a technique to evaluate predictive models by randomly partitioning the samples into k equal sized subsamples and repeated k times (K fold), with each of the subsamples used once as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. In C# accord.machinelearning.crossvalidation.dll, number of fold, type of learner, loss function, and inputs and outputs are defined, the training error and validation error can be computed.

The machine learning learners have default parameters and sometimes they are not optimized for the model. Accord has gridsearch.crossvalidate method which allows parameter tuning using grid search and cross-validation to find the best learner. In the following it shows that for c45 decisiontree learner, a grid search from 1 to 20 for “Join” parameter, and from 1 to 20 for “MaxHeight” parameters, with 3-fold cross-validation. “Join” means: “Gets or sets how many times one single variable can be integrated into the decision process. In the original ID3 algorithm, a variable can join only one time per decision path (path from the root to a leaf). If set to zero, a single variable can participate as many times as needed. Default is 1.” “Maxheight” means : “Gets or sets the maximum allowed height when learning a tree. If set to zero, the tree can have an arbitrary length. Default is 0.” The learner then apply the best parameter for model prediction. Be aware that grid search takes a long time to run depending on the range and the data size. So it should be run separately to find the parameters, then applied the parameters directly to the model. In the model implemented in IP, Join=3 and Maxheight=5 are used.

#region gridsearch crossvalidation to find best parameters

var gscv = GridSearch.CrossValidate(

// Here we can specify the range of the parameters to be included in the search

ranges: new

{

Join = GridSearch.Range(fromInclusive: 1, toExclusive: 20),

MaxHeight = GridSearch.Range(fromInclusive: 1, toExclusive: 20),

},

// Indicate how learning algorithms for the models should be created

learner: (p, ss) => new C45Learning

{

// Here, we can use the parameters we have specified above:

Join = p.Join,

MaxHeight = p.MaxHeight,

},

// Define how the model should be learned, if needed

fit: (teacher, x, y, w) => teacher.Learn(x, y, w),

// Define how the performance of the models should be measured

loss: (actual, expected, r) => new ZeroOneLoss(expected).Loss(actual),

folds: 3, // use k = 3 in k-fold cross validation

x: inputs, y: outputs // so the compiler can infer generic types

);

// If needed, control the parallelization degree

gscv.ParallelOptions.MaxDegreeOfParallelism = 1;

// Search for the best decision tree

var result = gscv.Learn(inputs, outputs);

// Get the best cross-validation result:

//var crossValidation = result.BestModel;

// Get an estimate of its error:

double bestAverageError = result.BestModelError;

double trainError = result.BestModel.Training.Mean;

double trainErrorVar = result.BestModel.Training.Variance;

double valError = result.BestModel.Validation.Mean;

double valErrorVar = result.BestModel.Validation.Variance;

// Get the best values for the parameters:

int bestJoin = result.BestParameters.Join;

int bestHeight = result.BestParameters.MaxHeight;

// Use the best parameter values to create the final

// model using all the training and validation data:

var bestTeacher = new C45Learning

{

Join = bestJoin,

MaxHeight = bestHeight,

};

// Use the best parameters to create the final tree model:

DecisionTree finalTree = bestTeacher.Learn(inputs, outputs);

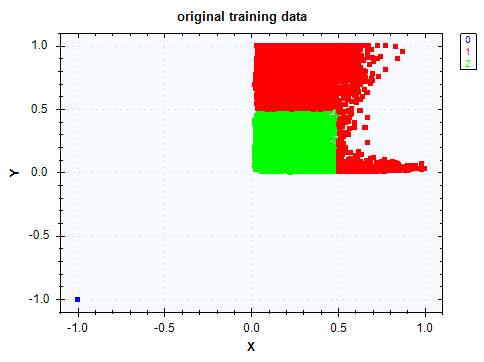
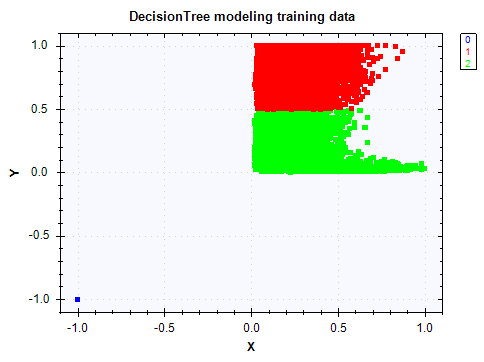
1. Building a classifier in IP( Interactive Petrophysics)

One application to do simple classifier is developed in IP. The app uses two curves as training dataset(by default gr and rdeep in this app), and then you can select any two curves as testing dataset or the curves of interest to be classified. Since the wells/database I tried does not have any labelled data, I created one within the code, and three labels are all saved as curves (the original label generated labelori, the training label labeltrain, and the label after applying the model called labeltest).

Decision tree classifier is applied with c45 learner because it can handle both continuous and discrete variables and here the features are continuous (c13 can only handle discrete variables).

The parameters of the learner have to be modified from the default one because without defining “Join” and “MaxHeight”, and leave them to default, the classification result was not right. The grid search for best parameters code are shown in section 4. It takes a long time to run the search, so it is suggested to be taken out after the one time search is done. The parameter is then hard coded.

The following figure shows the comparison of the before tuning, and it is definitely wrong. But still it gives an accuracy of 0.96. After tuning, it gives an accuracy of 1. Other metrics can be implemented in IP and displayed to provide performance evaluation.

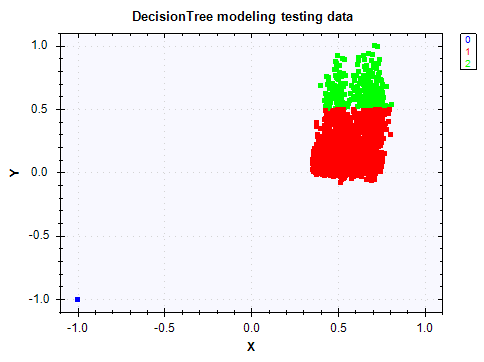
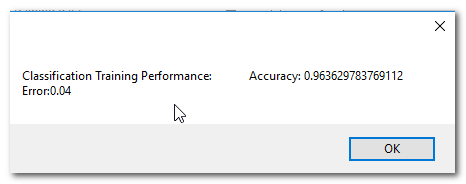
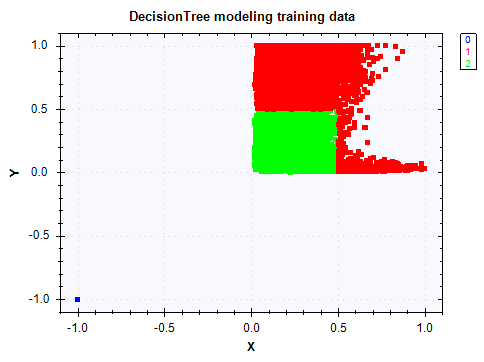
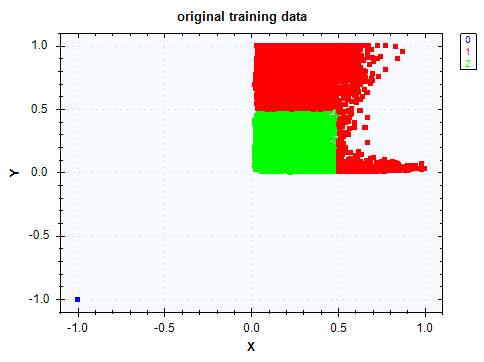
 

Fig. 11. The modeling without tuning, and the final result on test data.



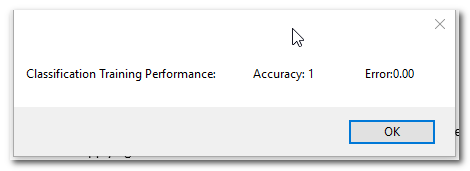
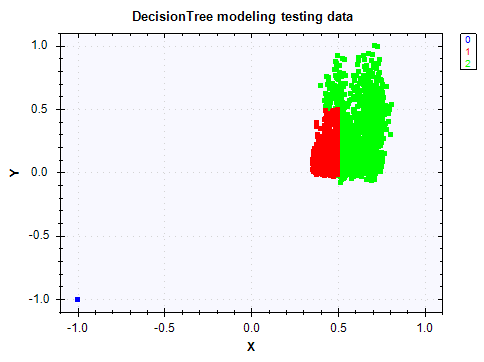


Fig. 12. The results after tuning: The original curves using gr and rdeep as training dataset, and the training model, and finally applying the model on test dataset DT and DRHO. Accuracy is 1.

The blue class is for NA values. They are put into one class so it will be easier to write back as a curve.

The input curves are not split into training-testing subsets for the model building, and it is can be implemented easily in the future.

The code can be modified to include more input curves, and the condition to generate a label, or one of the input as the label. It can use parameters input to enhance the features. It can be modified to include curves from different wells to implement the algorithm.

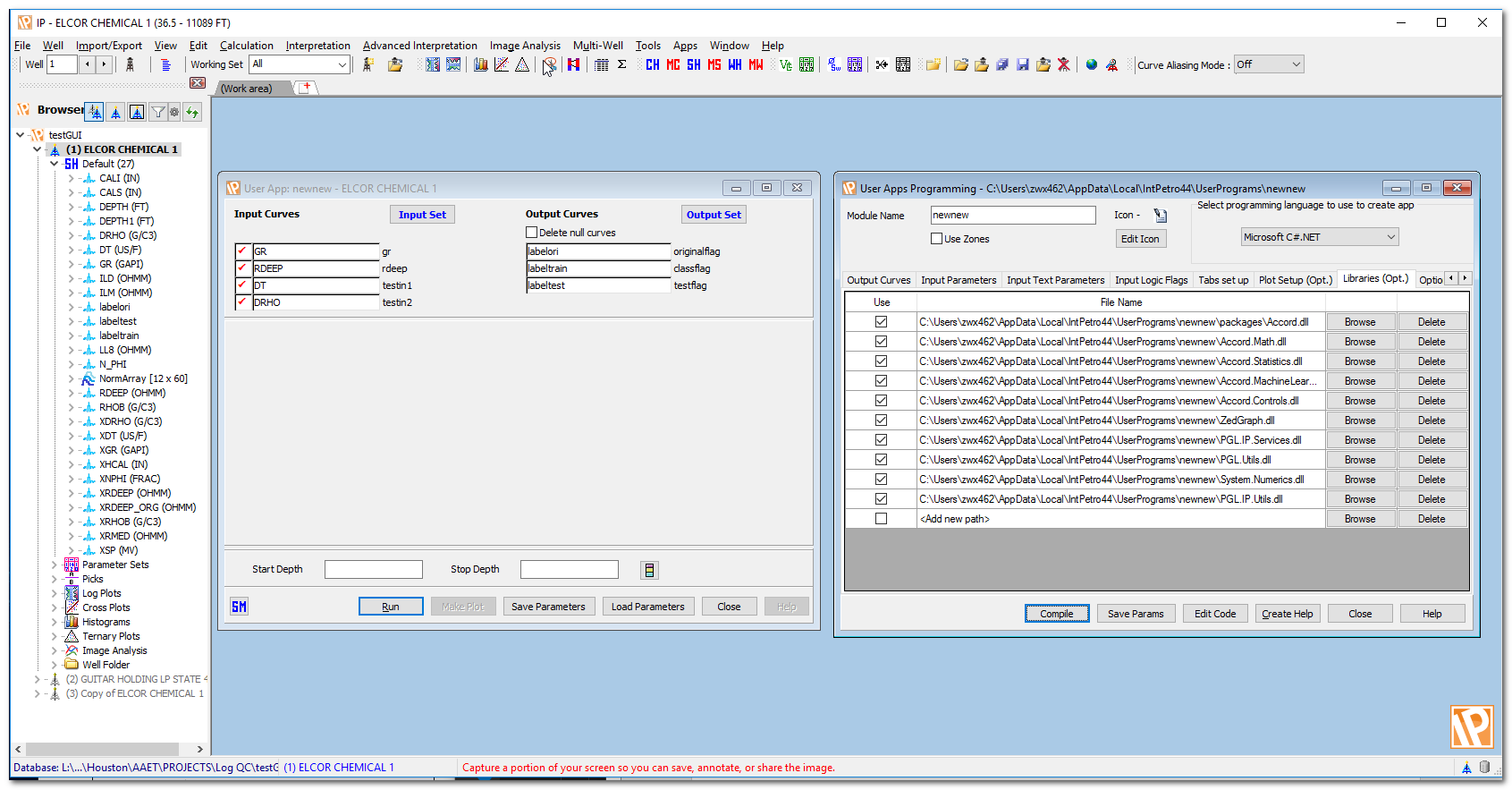


Fig. 13. The App diagram of input curves and output curves and the dynamic libraries it uses.

1. Future work and proposal

The supervised learning classification algorithms can be applied to well-log data. The demo here shows that they can be used to identify a binary class, for example, “Pay” or “No Pay” Zones. Multi-class problem can be tackled, for example, facies recognition. The training data would require pre-labeled class for training, then the model can be applied to automatically predict facies.

A simple classifier is built in IP for a quick demonstration of the idea and what algorithm can be used, and how to achieve the accurate classification by tuning the parameters of the learner. The resulted labels are then saved back as curves to be displayed.

The ideas can be extended to multiple input multiple output, and a neural network approach can be applied and the output is not limited to discrete labels but continuous ones too.