Finding Optimal Features for Credit Analysis

Vipul Munot
Indiana University
MS in Data Science
Indiana, Bloomington
vipmunot@iu.edu

Rahul Sampat
Indiana University
MS in Data Science
Indiana, Bloomington
rrsampat@umail.iu.edu

DATA SETS:

For this project, we are using the following 3 data sets from the UCI machine learning repository:

1) German Credit Data:

This is a binary classification problem, where based on the given set of attributes, a person is labelled as a good(1)/bad(2) credit risk. There are a total of 1000 instances with 20 attributes and a label.
URL: https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)
Below is the attribute description:

Attribute	**Type**	**Description**	**Example**
Attribute 1	Qualitative	Status of existing checking account	A11 : ... < 0 DM A12 : 0 <= ... < 200 DM A13 : ... >= 200 DM / salary assignments for at least 1 year A14 : no checking account
Attribute 2	Numerical	Duration in month	6, 48 etc
Attribute 3	Qualitative	Credit history	A30 : no credits taken/ all credits paid back duly A31 : all credits at this bank paid back duly A32 : existing credits paid back duly till now A33 : delay in paying off in the past A34 : critical account/ other credits existing (not at this bank)
Attribute 4	Qualitative	Purpose	A40 : car (new) A41 : car (used) A42 : furniture/equipment A43 : radio/television A44 : domestic appliances A45 : repairs A46 : education A47 : (vacation - does not exist?) A48 : retraining A49 : business A410 : others
Attribute 5	Numerical	Credit amount	1169, 5951 etc.
Attribute 6	Qualitative	Savings account/bonds	A61 : ... < 100 DM A62 : 100 <= ... < 500 DM A63 : 500 <= ... < 1000 DM

			A64 : .. >= 1000 DM A65 : unknown/ no savings account
Attribute 7	Qualitative	Present employment since	A71 : unemployed A72 : ... < 1 year A73 : 1 <= ... < 4 years A74 : 4 <= ... < 7 years A75 : .. >= 7 years
Attribute 8	Numerical	Installment rate in percentage of disposable income	1,2,3,4 etc.
Attribute 9	Qualitative	Personal status and sex	A91 : male : divorced/separated A92 : female : divorced/separated/married A93 : male : single A94 : male : married/widowed A95 : female : single
Attribute 10	Qualitative	Other debtors / guarantors	A101 : none A102 : co-applicant A103 : guarantor
Attribute 11	Numerical	Present residence since	1,2,3,4 etc.
Attribute 12	Qualitative	Property	A121 : real estate A122 : if not A121 : building society savings agreement/ life insurance A123 : if not A121/A122 : car or other, not in attribute 6 A124 : unknown / no property
Attribute 13	Numerical	Age in years	22, 49 etc.
Attribute 14	Qualitative	Other installment plans	A141 : bank A142 : stores A143 : none
Attribute 15	Qualitative	Housing	A151 : rent A152 : own A153 : for free
Attribute 16	Numerical	Number of existing credits at this bank	1,2,3 etc.
Attribute 17	Qualitative	Job	A171 : unemployed/ unskilled - non-resident A172 : unskilled - resident A173 : skilled employee / official A174 : management/ self-employed/highly qualified employee/ officer
Attribute 18	Numerical	Number of people being liable to provide maintenance for	1,2
Attribute 19	Qualitative	Telephone	A191 : none

			A192 : yes, registered under the customer's name
Attribute 20	Qualitative	Foreign Worker	A201 : yes A202 : no
Label	Binary	Indicates good/bad risk	1 = Good, 2 = Bad

Note: For this dataset, it is worse to class a customer as good when they are bad (5), than it is to class a customer as bad when they are good (1).

2) Credit Approval Data Set:

This data set concerns credit card applications.

This is a binary classification problem, where based on the given set of attributes, a person is labelled as a + (positive)/ - (negative) candidate for issuing a credit card. There are a total of 690 instances with 15 attributes and a label.

This dataset is interesting because there is a good mix of attributes -- continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values.

URL: https://archive.ics.uci.edu/ml/datasets/Credit+Approval

Below is the attribute description:

Attribute	**Type/Set of values**
A1	b,a.
A2	Continuous
A3	Continuous
A4	u, y, l, t
A5	g, p, gg
A6	c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.
A7	v, h, bb, j, n, z, dd, ff, o.
A8	Continuous
A9	t, f.
A10	t, f.
A11	Continuous
A12	t,f
A13	g,p,s
A14	Continuous
A15	Continuous
A16	+,-

Missing Attribute Values:

37 cases (5%) have one or more missing values.

Class Distribution:

+: 307 (44.5%)

-: 383 (55.5%)

3) Default of Credit Card Clients Data Set:

This research aimed at the case of customers default payments in Taiwan.

From the perspective of risk management, the result of predictive accuracy of the estimated probability of default will be more valuable than the binary result of classification - credible or not credible clients.

This research employed a binary variable, default payment (Yes = 1, No = 0), as the response variable

There are a total of 30000 instances with 23 attributes and a label.

URL: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

Below is the attribute description:

Attribute	**Description**	**Examples**
X1	Amount of the given credit (NT dollar)	It includes both the individual consumer credit and his/her family (supplementary) credit.
X2	Gender	(1 = male; 2 = female).
X3	Education	(1 = graduate school; 2 = university; 3 = high school; 4 = others)
X4	Marital status	(1 = married; 2 = single; 3 = others)
X5	Age	Year
X6 - X11	History of past payment. We tracked the past monthly payment records (from April to September, 2005)	X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005; . . .; X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; …and so on;
X12-X17	Amount of bill statement (NT dollar)	X12 = amount of bill statement in September, 2005; X13 = amount of bill statement in August, 2005; X17 = amount of bill statement in April, 2005.
X18-X23	Amount of previous payment (NT dollar).	X18 = amount paid in September, 2005; X19 = amount paid in August, 2005; . . .; X23 = amount paid in April, 2005
Y	Label	Default payment (Yes = 1, No = 0),

METHOD:

Data Splitting:

We used 90-10% train-test split of data.

Machine Learning Techniques:

We self implemented 2 algorithms and used scikit libraries for 5 algorithms, on all the 3 datasets above:

Self Implemented:

a) Naïve Bayes
b) kNN

Using Scikit Libraries:

a) Naive Bayes
b) Random Forest
c) Logistic Regression
d) Support Vector Machines
e) K Nearest Neighbors

Parameter Settings for Sci-kit Packages:

We tried out various parameter tuning settings to find out the best ones, based on a greedy approach, ie. changing one parameter at a time, for each of the algorithm on each of the data set seperately. You may see the output of these parameter setting by executing final.py file provided. Below are the best parameter settings for each of the algorithms on each dataset:

a) Random Forest

In Random Forest, we hyper tuned the parameters according to area under ROC curve and the accuracy. The parameters we tuned are max_depth, max_features,n_estimators,random_state and min_samples_leaf. Following are the final parameters settings we used to maximize the accuracy

Dataset: German Dataset

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini', max_depth=800, max_features='auto', max_leaf_nodes=None, min_impurity_split=1e-07, min_samples_leaf=50, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=600, n_jobs=-1, oob_score=False, random_state=None, verbose=0, warm_start=False)

Dataset: Credit Approval Data Set

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',max_depth=300, max_features='auto', max_leaf_nodes=None, min_impurity_split=1e-07, min_samples_leaf=1,min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=800, n_jobs=-1, oob_score=False, random_state=100,verbose=0, warm_start=False)

Dataset: Default of Credit Card Clients Data Set

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini', max_depth=100, max_features='auto', max_leaf_nodes=None, min_impurity_split=1e-07, min_samples_leaf=50, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=600, n_jobs=-1, oob_score=False, random_state=None, verbose=0, warm_start=False)

b) Logistic Regression

In Logistic Regression, we hyper tuned the parameters according to area under ROC curve and the accuracy. The parameters we tuned are penalty, solver, C, class_weight, max_iter and random_state. Following are the final parameters settings we used to maximize the accuracy.

Dataset: German Dataset
LogisticRegression(C=0.5, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=-1, penalty='l2', random_state=500, solver='liblinear', tol=0.0001, verbose=0, warm_start=False)

Dataset: Credit Approval Data Set
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=-1, penalty='l2', random_state=None, solver='liblinear', tol=0.0001, verbose=0, warm_start=False)

Dataset: Default of Credit Card Clients Data Set
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=-1, penalty='l2', random_state=None, solver='liblinear', tol=0.0001, verbose=0, warm_start=False)

c) kNN:

In k Nearest Neighbors, we hyper tuned the parameters according to area under ROC curve and the accuracy. The parameters we tuned are n_neighbors, weights and algorithm. Following are the final parameters settings we used to maximize the accuracy.

Dataset: German Dataset
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=-1, n_neighbors=5, p=2, weights='uniform')

Dataset: Credit Approval Data Set
KNeighborsClassifier(algorithm='ball_tree', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=-1, n_neighbors=10, p=2, weights='uniform')

Dataset: Credit Approval Data Set
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',metric_params=None, n_jobs=-1, n_neighbors=50, p=2, weights='uniform')

d) Linear SVC:

In Linear Support Vector Machine, we hyper tuned the parameters according to area under ROC curve and the accuracy. The parameters we tuned are dual,C, class_weight, penalty and random_state. Following are the final parameters settings we used to maximize the accuracy.

Dataset: German Dataset
LinearSVC(C=1, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, loss='squared_hinge', max_iter=1000, multi_class='ovr', penalty='l1', random_state=1000, tol=0.0001, verbose=0)

Dataset: Credit Approval Data Set
LinearSVC(C=1, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, loss='squared_hinge', max_iter=500, multi_class='ovr', penalty='l2', random_state=1000, tol=0.0001,verbose=0)

Dataset: Credit Approval Data Set
LinearSVC(C=1, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, loss='squared_hinge', max_iter=10, multi_class='ovr', penalty='l1', random_state=1000, tol=0.0001, verbose=0)

e) Naïve Bayes:

There are no parameter to be set for NB.

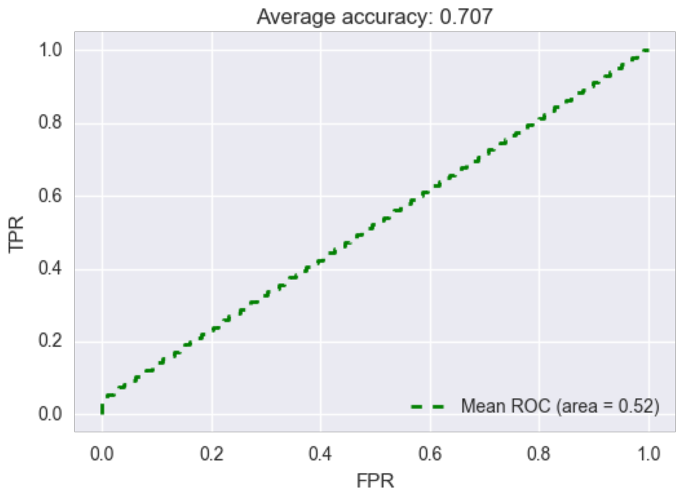
Dataset: German Dataset/ Credit Approval Data Set/ Credit Approval Data Set

GaussianNB()

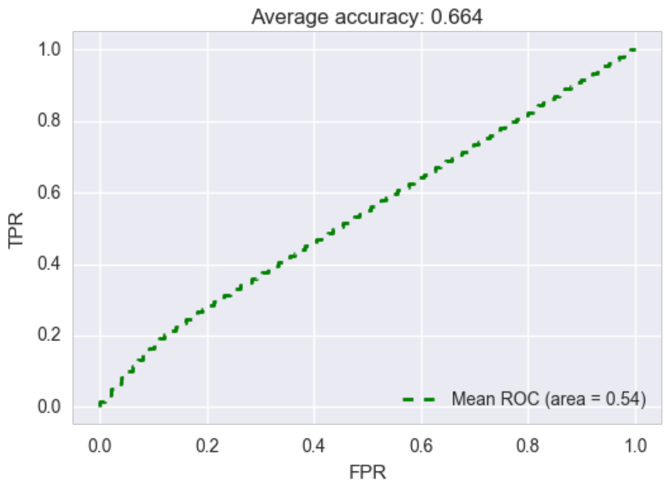
ROC Curves:

The average ROC (over 10-fold cross validation) for different algorithms over different datasets are as follows:

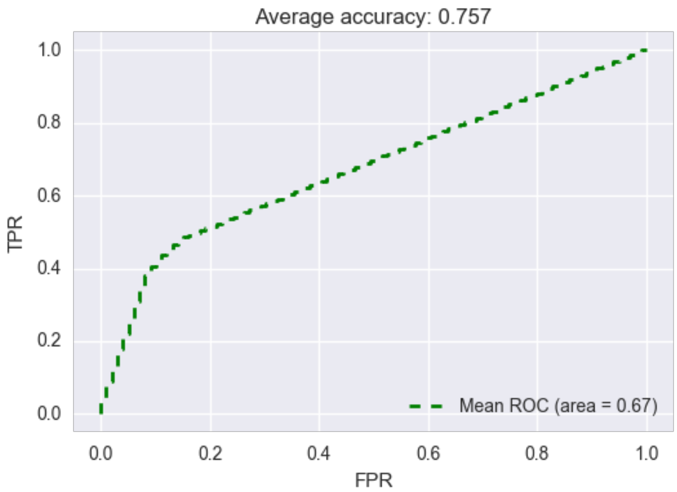
1) German Dataset:



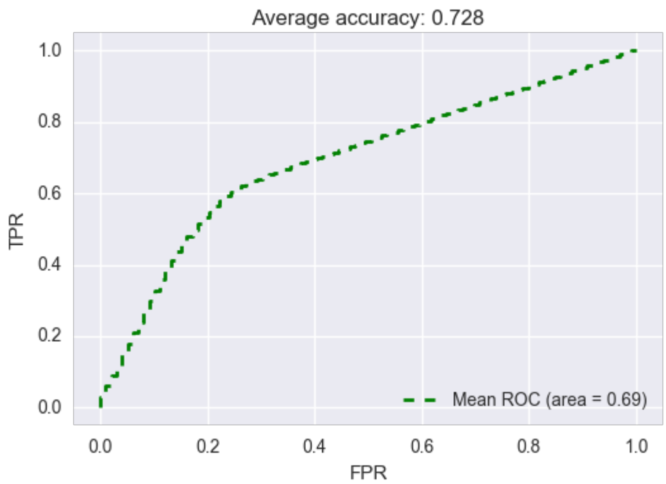
Random Forest



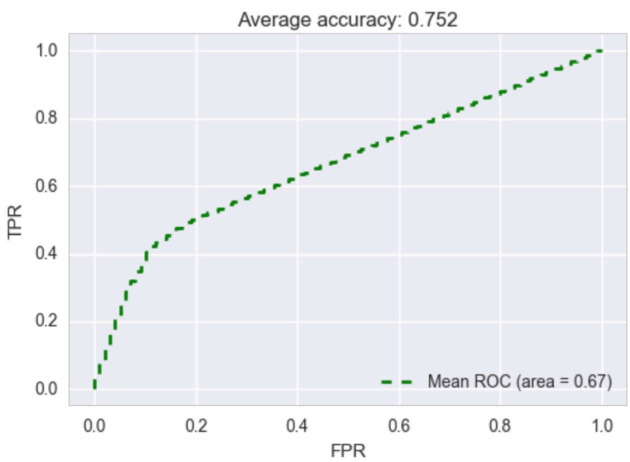
kNN



Logistic Regression

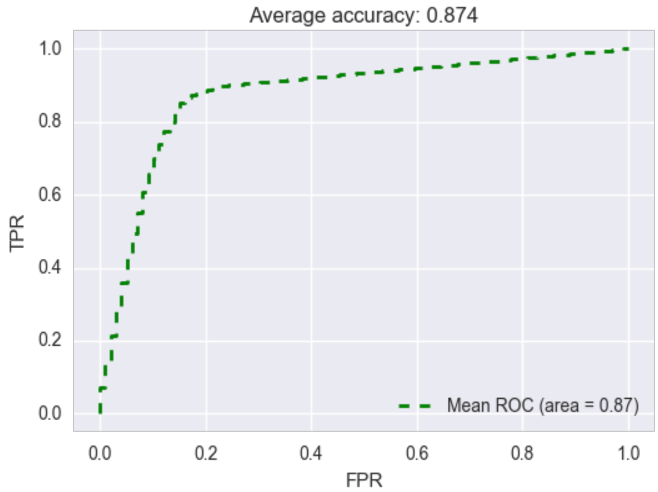


Naïve Bayes

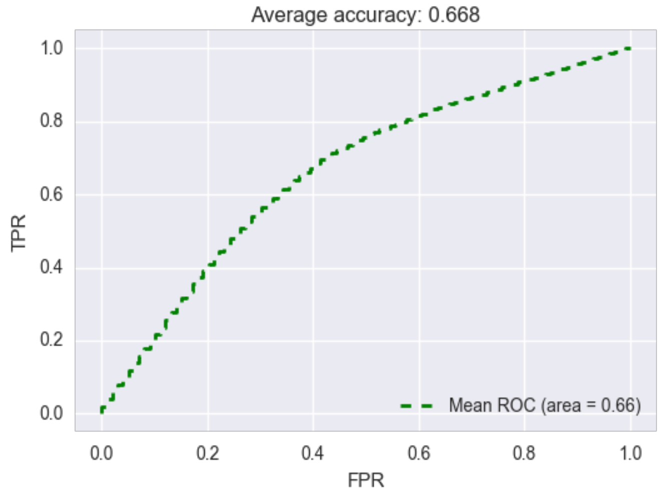


SVM

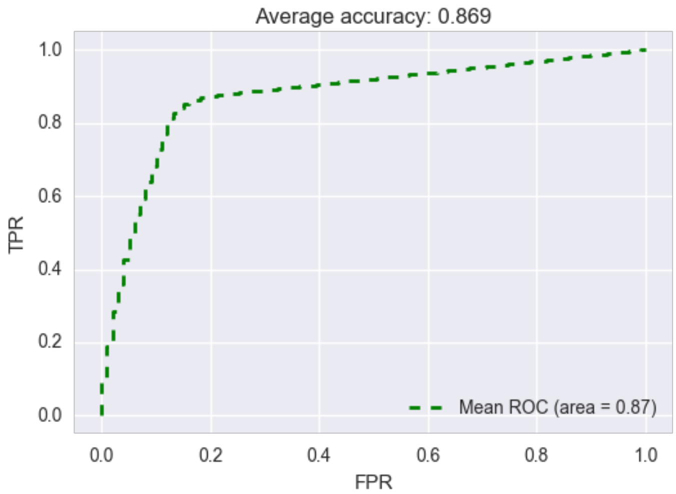
2) Credit Approval Data Set:



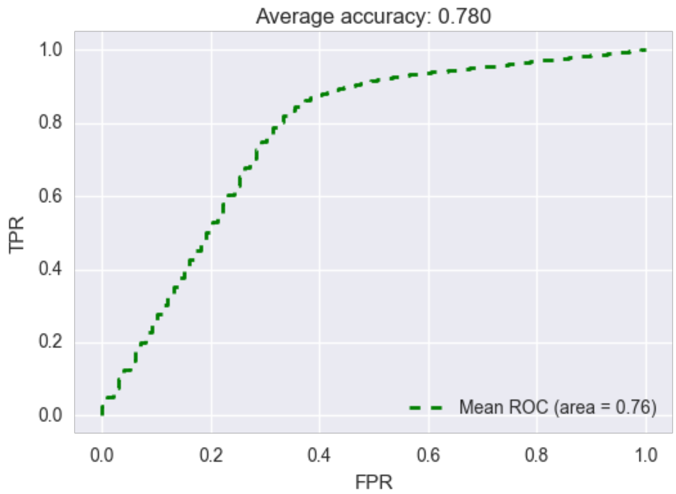
Random Forest



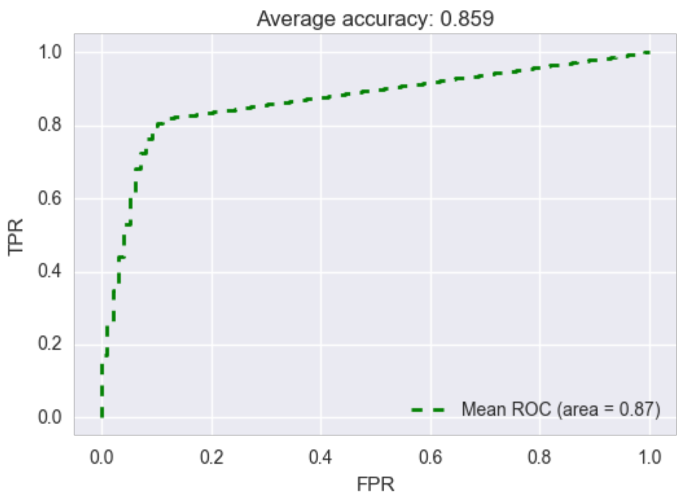
kNN



Logistic Regression

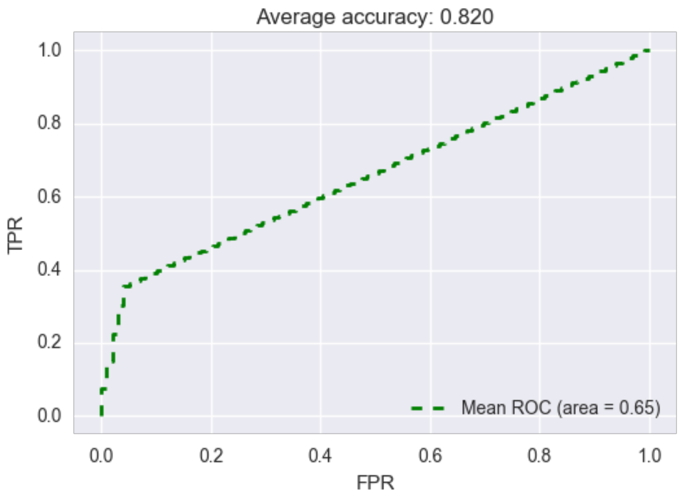


Naïve Bayes

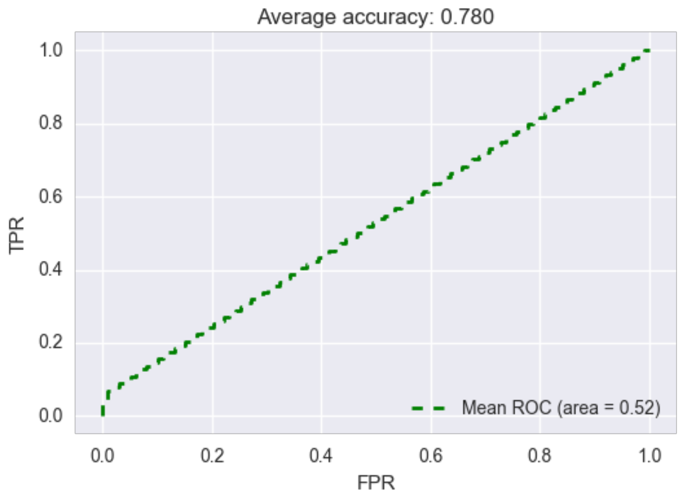


SVM

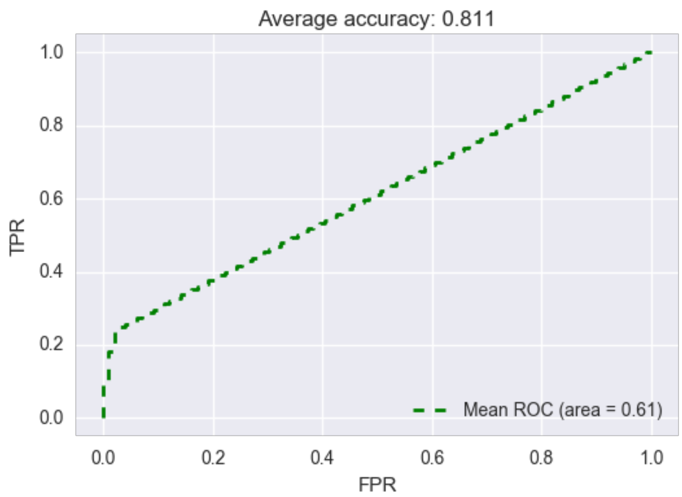
3) Default of Credit Card Clients Data Set:



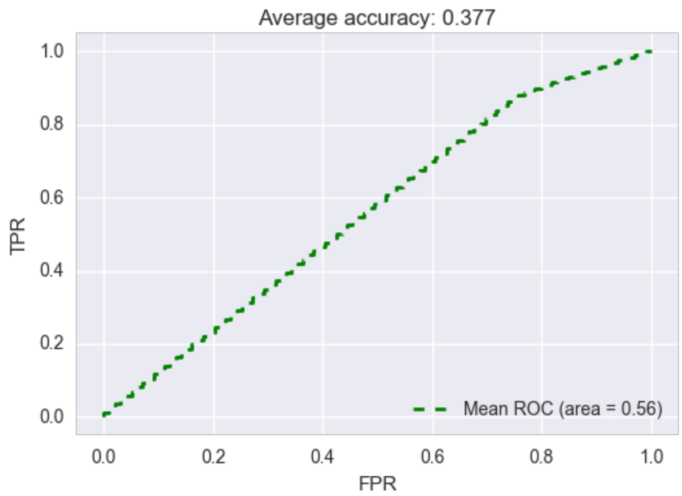
Random Forest



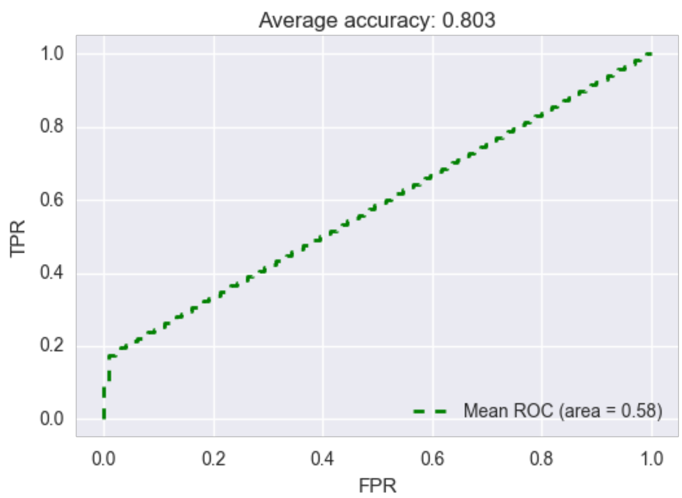
kNN



Logistic Regression



Naïve Bayes



SVM

Confusion Matrices:

Below are the confusion matrices using self implemented and scikit libraires with 0.1-test/0.9-train data split:

1) Self-Implemented:

a) Naïve Bayes:

German Dataset:

Confusion Matrix

Model Results

Actual\Model	1	2	Actual Count
1	55	21	76
2	10	21	31

('Naive Bayes Accuracy', 71.02803738317758)

CRX Dataset:

Confusion Matrix

Model Results

Actual\Model	+	-	Actual Count
+	23	9	32
-	8	33	41

('Naive Bayes Accuracy', 76.71232876712328)

Default Dataset:

Confusion Matrix

Model Results

Actual\Model	1	0	Actual Count
1	275	415	690
0	362	1941	2303

('Naive Bayes Accuracy', 74.0394253257601)

b) kNN:

German Dataset:

Accuracy_score: 68.95%

CRX Dataset:

Accuracy_score: 65.94%

Default Dataset:

Accuracy_score: 75.5%

2) Scikit:

a) German Dataset:

Random Forest

Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	282	5
Actual No	18	695

Accuracy : 70%
True Positive Rate: 98.25%
False Positive Rate: 2.52%

Logistic

Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	163	80
Actual No	137	620

Accuracy 75.7 %
True Positive Rate: 67.07%
False Positive Rate: 18.09%

Naïve Bayes

Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	118	146
Actual No	182	554

Accuracy 72.8%
True Positive Rate: 44.69%
False Positive Rate: 24.72%

kNN

Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	231	107
Actual No	169	593

Accuracy 66.4%
True Positive Rate: 68.3%
False Positive Rate: 10.42%

SVM

Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	163	82
Actual No	137	618

Accuracy 75.2%
True Positive Rate: 66.53%
False Positive Rate: 18.14%

b) Credit Approval (CRX) Dataset:

Random Forest
Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	37	40
Actual No	346	267

Accuracy 87.4%
True Positive Rate: 48.05%
False Positive Rate: 56.44%

Logistic
Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	56	32
Actual No	327	275

Accuracy 86.9%
True Positive Rate: 63.63%
False Positive Rate: 54.31%

Naïve Bayes
Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	35	112
Actual No	348	195

Accuracy 78%
True Positive Rate: 23.80%
False Positive Rate: 64.08%

kNN
Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	107	119
Actual No	276	188

Accuracy 66.8%
True Positive Rate: 47.34%
False Positive Rate: 59.48%

SVM
Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	73	27
Actual No	310	280

Accuracy 85.9%
True Positive Rate: 66.53%
False Positive Rate: 18.14%

c) Default Dataset:

Random Forest
Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	4266	1124
Actual No	2370	22240

Accuracy 82%
True Positive Rate: 79.14%
False Positive Rate: 9.63%

Logistic
Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	5043	638
Actual No	1593	22726

Accuracy 81.1%
True Positive Rate: 88.76%
False Positive Rate: 6.55%

Naïve Bayes
Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	771	17889
Actual No	5865	5475

Accuracy 37.7%
True Positive Rate: 4.13%
False Positive Rate: 51.71%

kNN
Confusion Matrix

	Predicted Yes	Predicted No
Actual Yes	6220	410
Actual No	416	22954

Accuracy 78%
True Positive Rate: 93.81%
False Positive Rate: 1.78%

SVM
Confusion Matrix

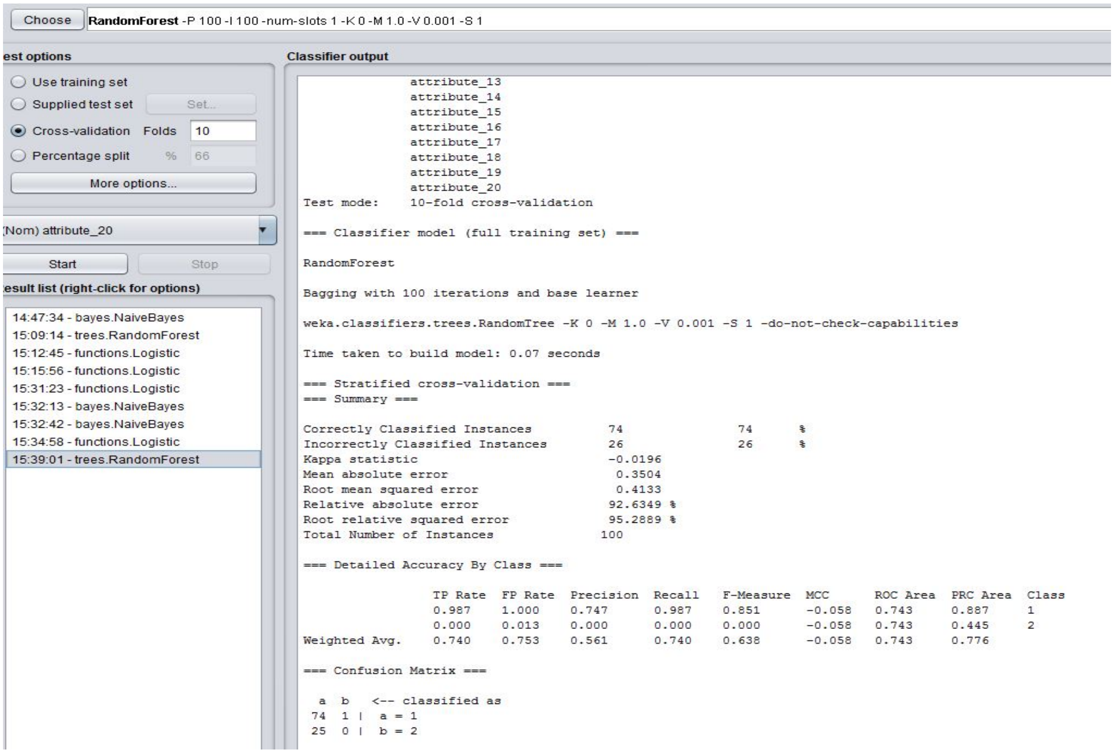
	Predicted Yes	Predicted No
Actual Yes	5455	438
Actual No	1181	22926

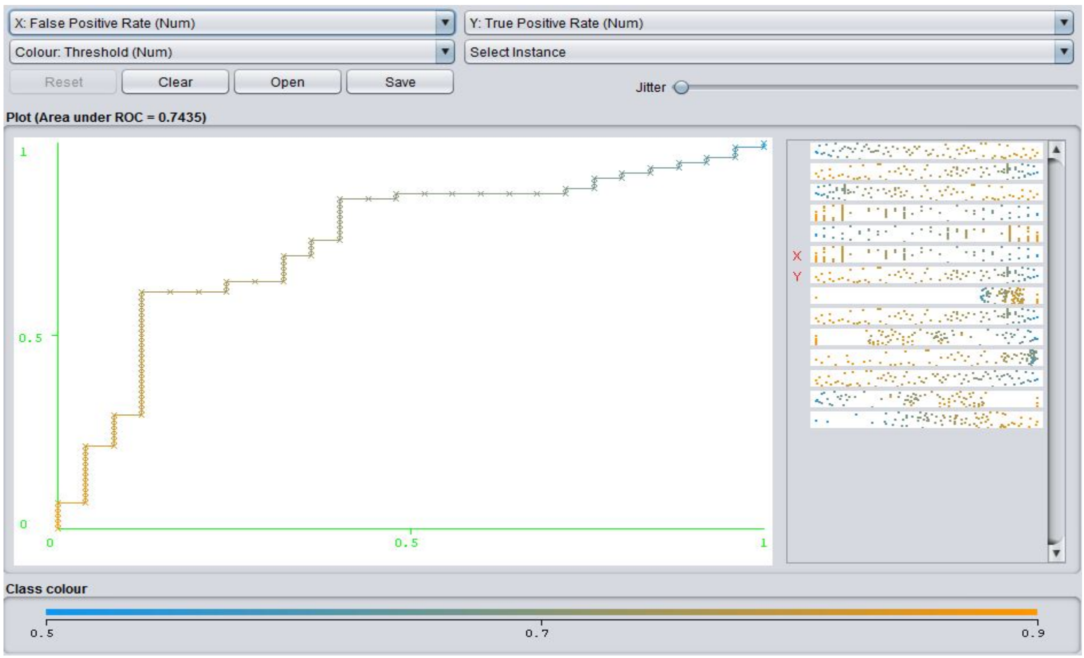
Accuracy 80.3%
True Positive Rate: 92.56%
False Positive Rate: 4.89%

WEKA ANALYSIS:

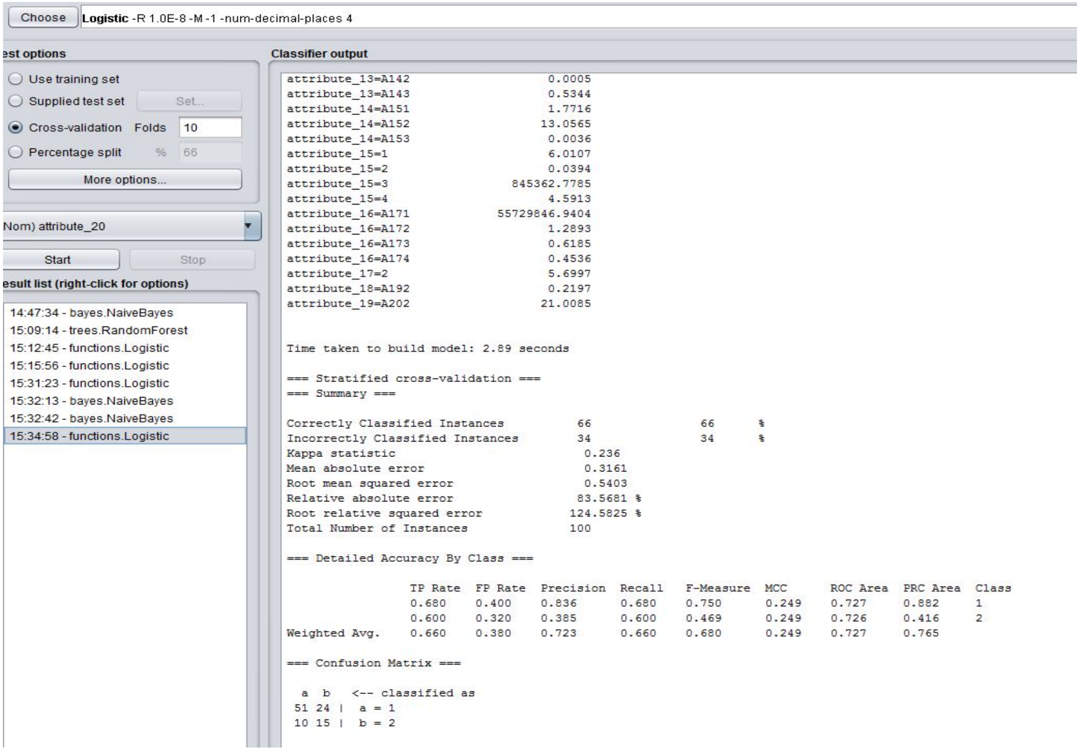
a) German Dataset:

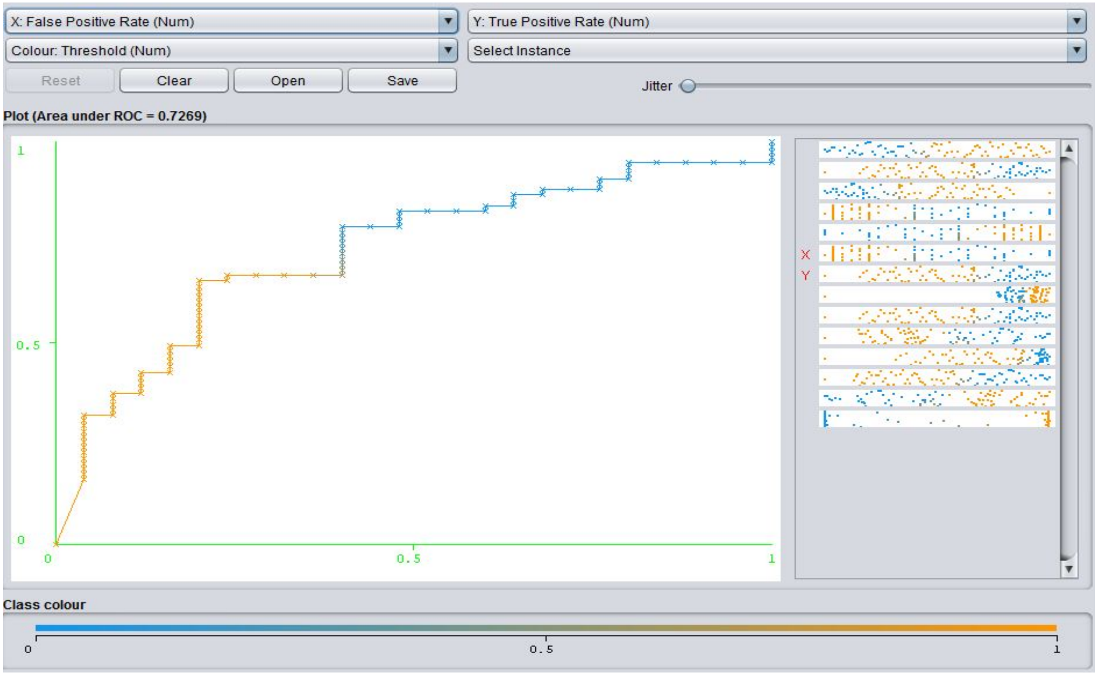
Random Forest:



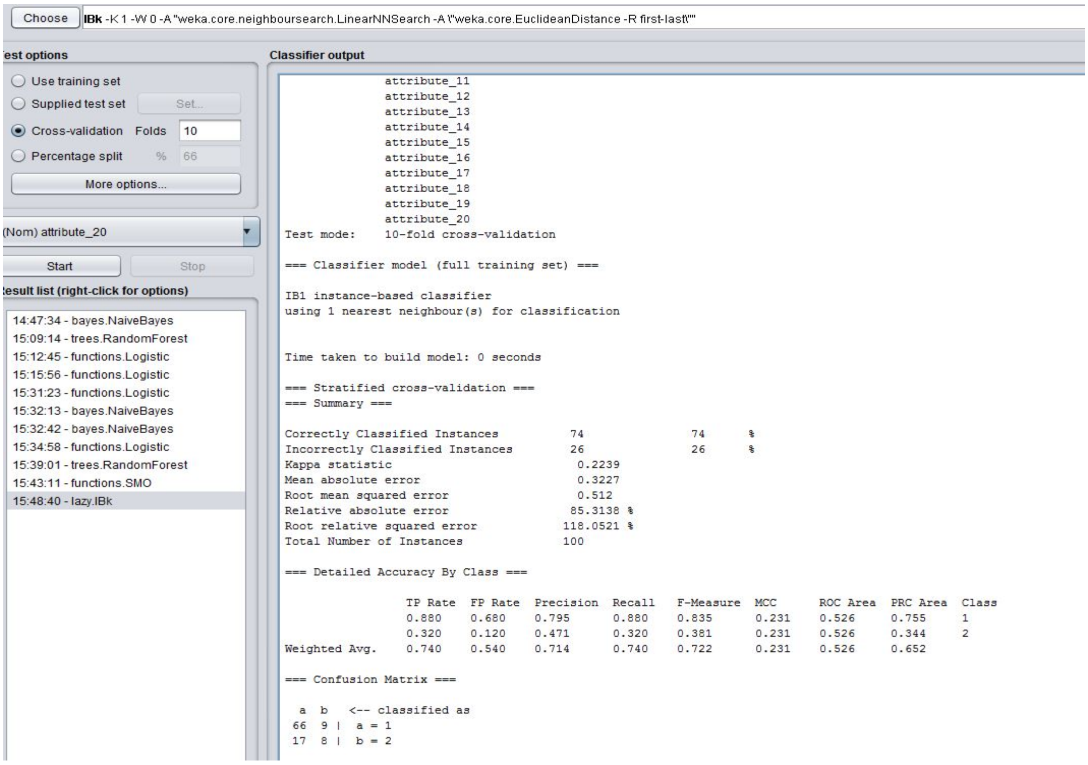


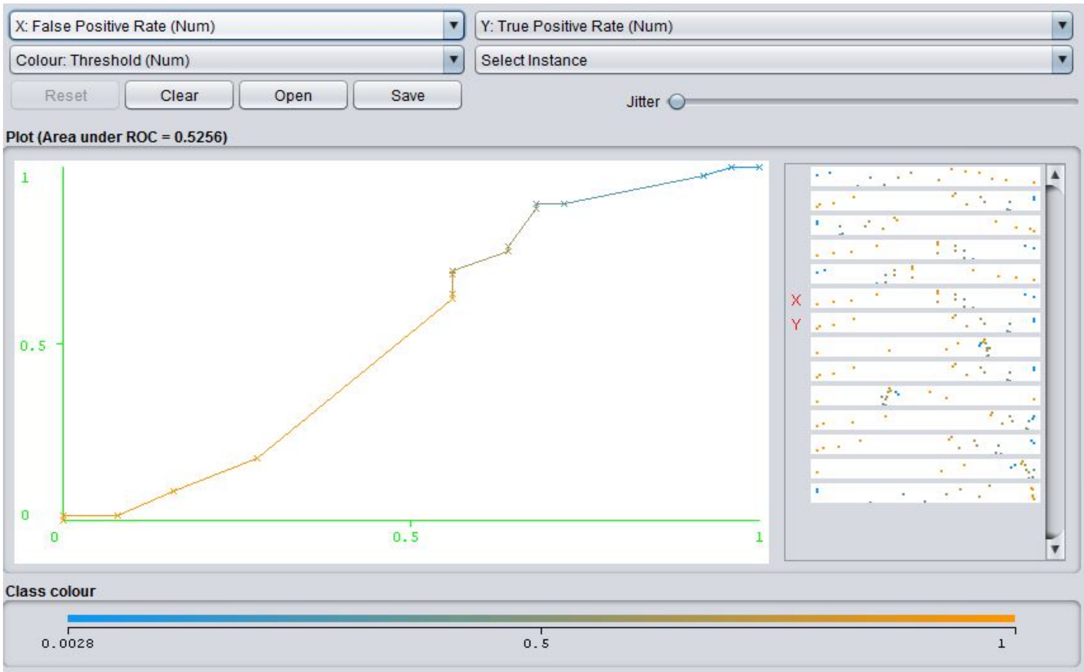
Logistic Regresion:



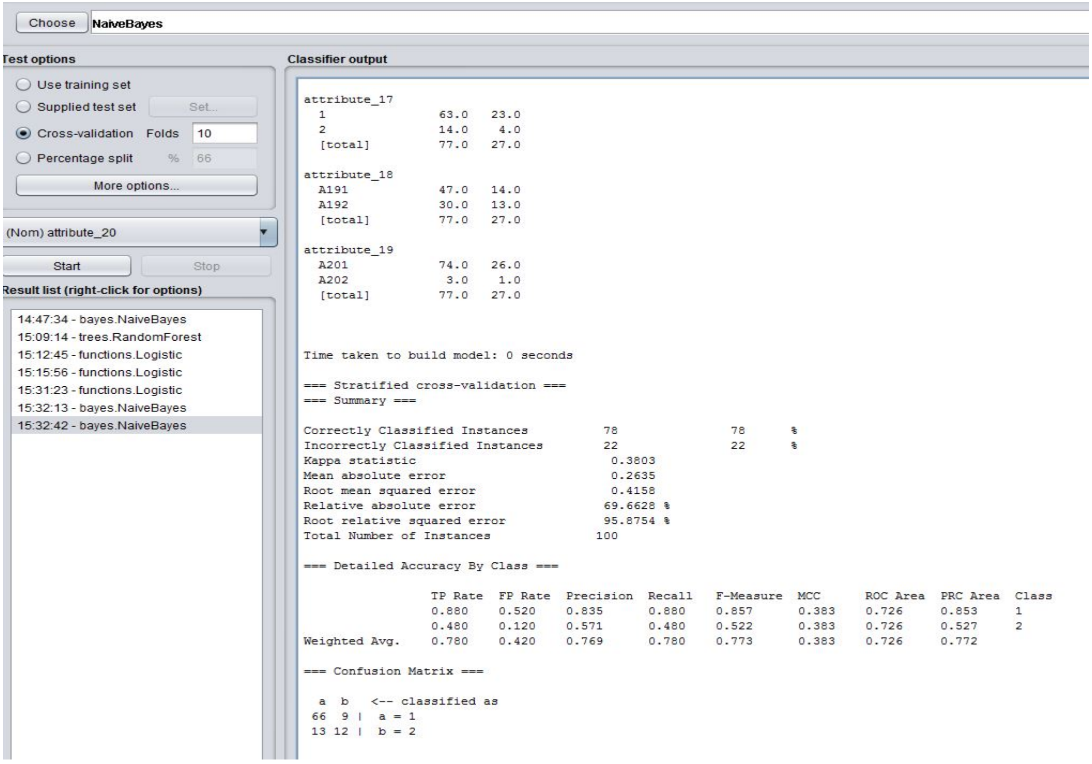


kNN:

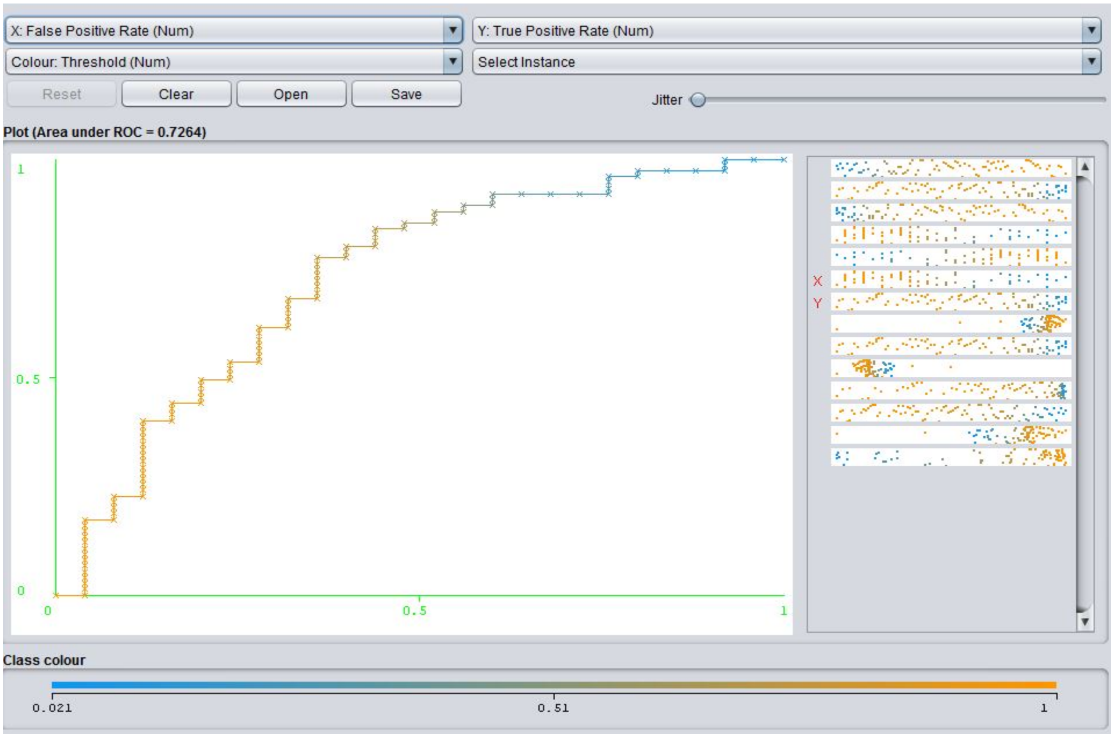

Choose
IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
est options
Use training set
Supplied test set
Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) attribute_20
Start
Stop
esult list (right-click for options)
14:47:34 - bayes.NaiveBayes
15:09:14 - trees.RandomForest
15:12:45 - functions.Logistic
15:15:56 - functions.Logistic
15:31:23 - functions.Logistic
15:32:13 - bayes.NaiveBayes
15:32:42 - bayes.NaiveBayes
15:34:58 - functions.Logistic
15:39:01 - trees.RandomForest
15:43:11 - functions.SMO
15:48:40 - lazy.IBk
Classifier output
attribute_11
attribute_12
attribute_13
attribute_14
attribute_15
attribute_16
attribute_17
attribute_18
attribute_19
attribute_20
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
IB1 instance-based classifier
using 1 nearest neighbour(s) for classification
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 74 74 %
Incorrectly Classified Instances 26 26 %
Kappa statistic 0.2239
Mean absolute error 0.3227
Root mean squared error 0.512
Relative absolute error 85.3138 %
Root relative squared error 118.0521 %
Total Number of Instances 100
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.880 0.680 0.795 0.880 0.835 0.231 0.526 0.755 1
0.320 0.120 0.471 0.320 0.381 0.231 0.526 0.344 2
Weighted Avg. 0.740 0.540 0.714 0.740 0.722 0.231 0.526 0.652
=== Confusion Matrix ===
a b <-- classified as
66 9 | a = 1
17 8 | b = 2


X: False Positive Rate (Num)
Y: True Positive Rate (Num)
Colour: Threshold (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot (Area under ROC = 0.5256)
1
0.5
0
0
0.5
1
X
Y
Class colour
0.0028
0.5
1

Naïve Bayes:

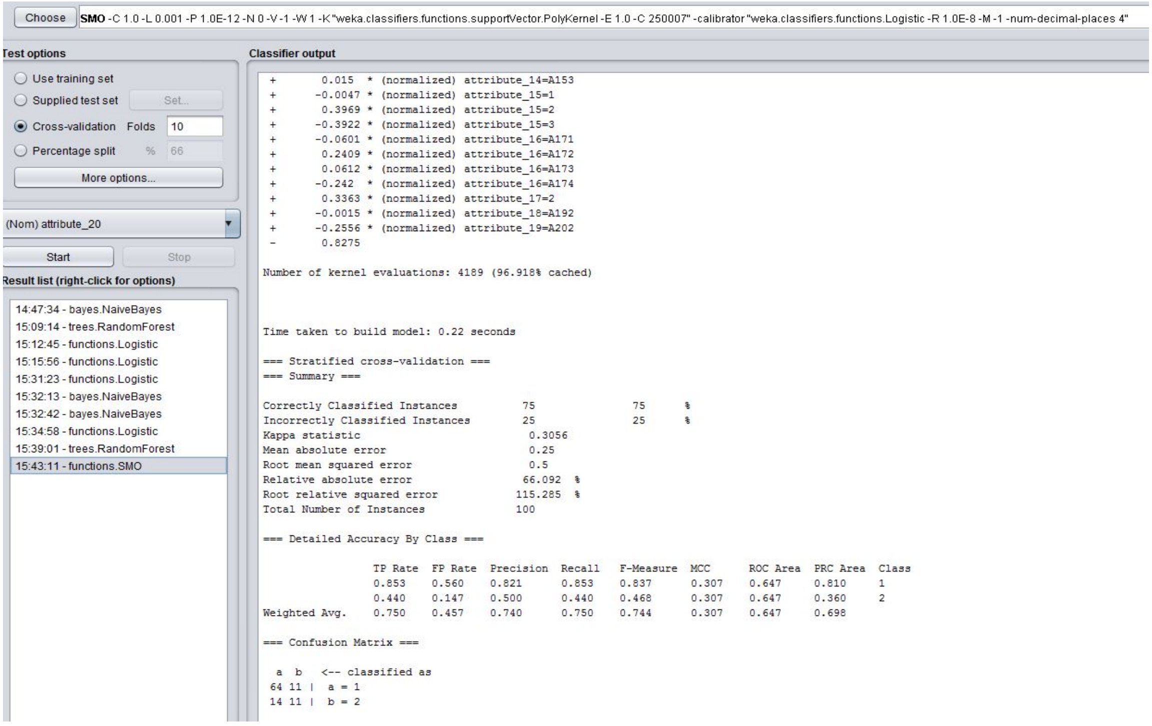


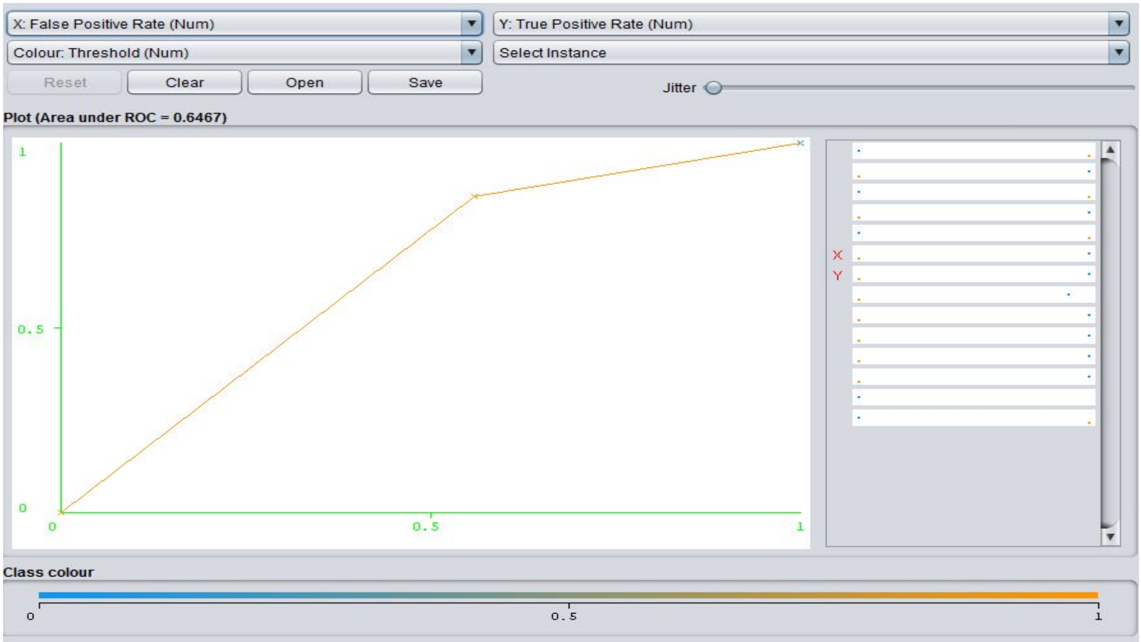
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) attribute_20
Start
Stop
Result list (right-click for options)
14:47:34 - bayes.NaiveBayes
15:09:14 - trees.RandomForest
15:12:45 - functions.Logistic
15:15:56 - functions.Logistic
15:31:23 - functions.Logistic
15:32:13 - bayes.NaiveBayes
15:32:42 - bayes.NaiveBayes
Classifier output
attribute_17
1 63.0 23.0
2 14.0 4.0
[total] 77.0 27.0
attribute_18
A191 47.0 14.0
A192 30.0 13.0
[total] 77.0 27.0
attribute_19
A201 74.0 26.0
A202 3.0 1.0
[total] 77.0 27.0
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 78 78 %
Incorrectly Classified Instances 22 22 %
Kappa statistic 0.3803
Mean absolute error 0.2635
Root mean squared error 0.4158
Relative absolute error 69.6628 %
Root relative squared error 95.8754 %
Total Number of Instances 100
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.880 0.520 0.835 0.880 0.857 0.383 0.726 0.853 1
0.480 0.120 0.571 0.480 0.522 0.383 0.726 0.527 2
Weighted Avg. 0.780 0.420 0.769 0.780 0.773 0.383 0.726 0.772
=== Confusion Matrix ===
a b <-- classified as
66 9 | a = 1
13 12 | b = 2



X: False Positive Rate (Num)
Y: True Positive Rate (Num)
Colour: Threshold (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot (Area under ROC = 0.7264)
1
0.5
0
0
0.5
1
X
Y
Class colour
0.021
0.51
1

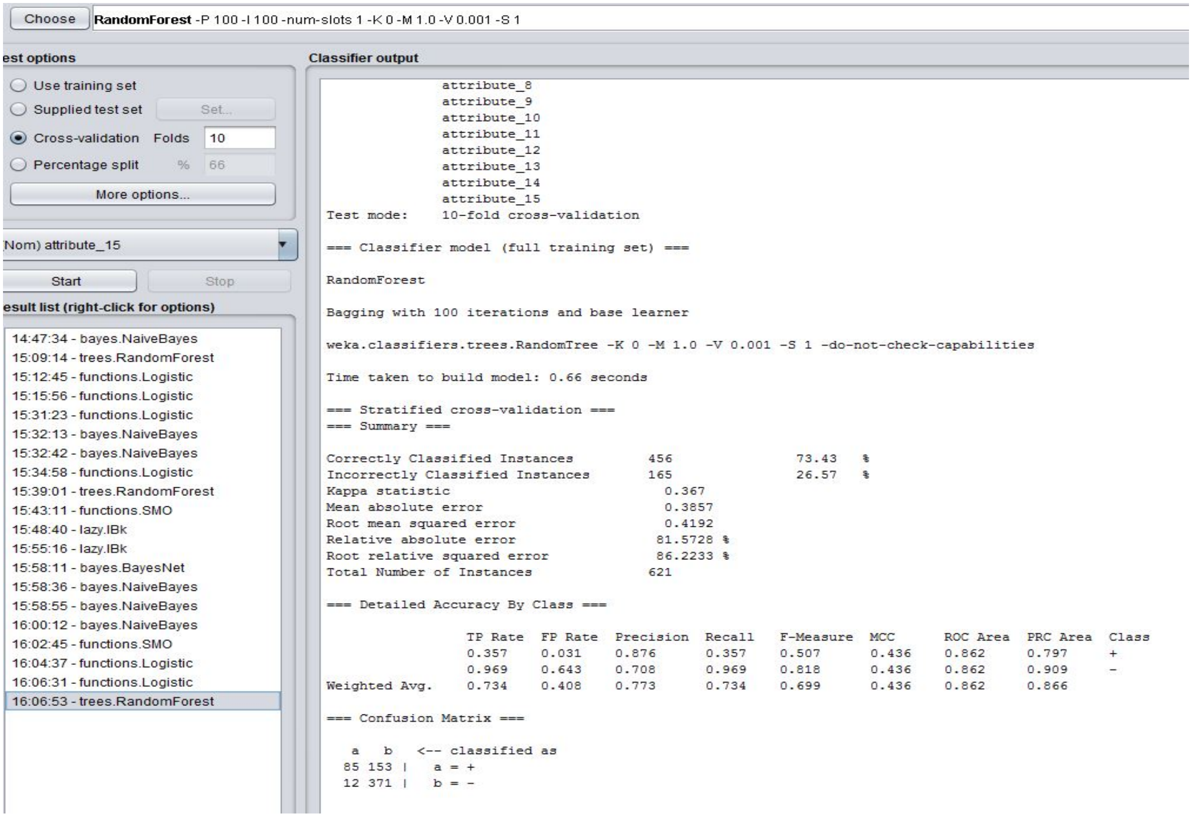
Support Vector Machine (Poly):

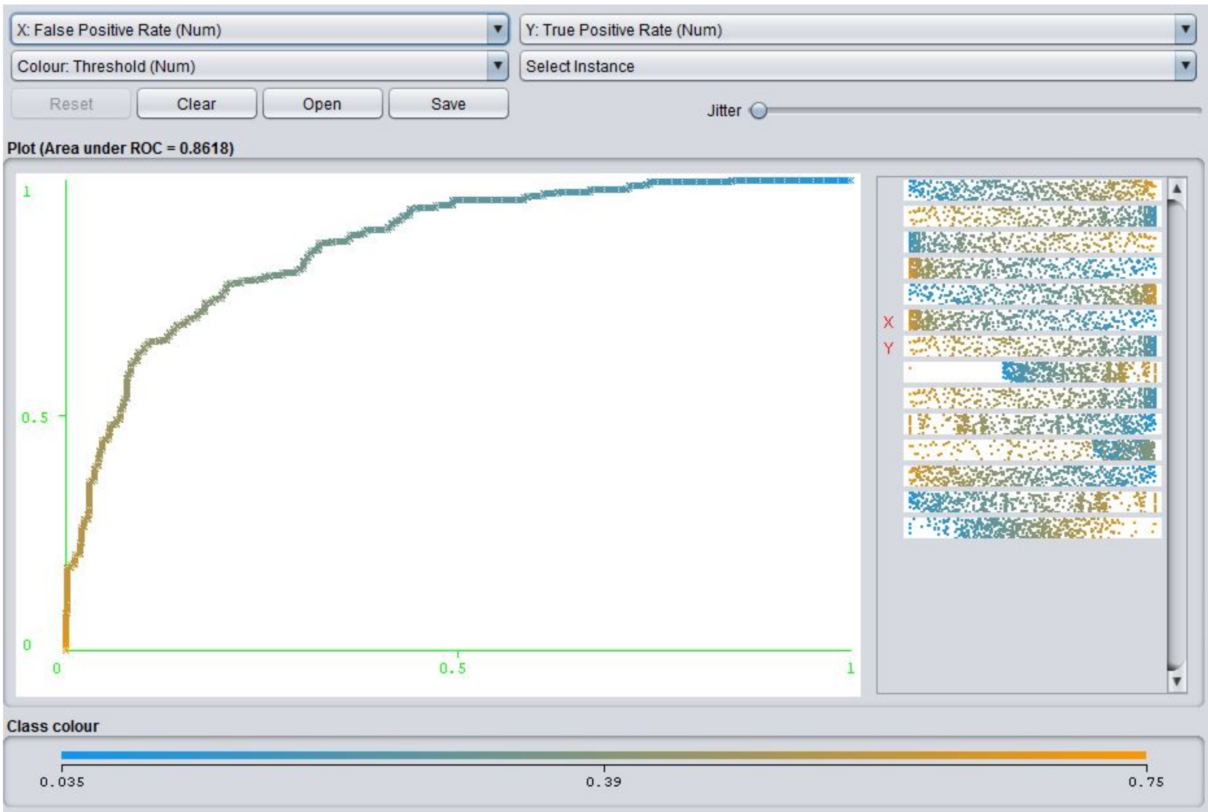




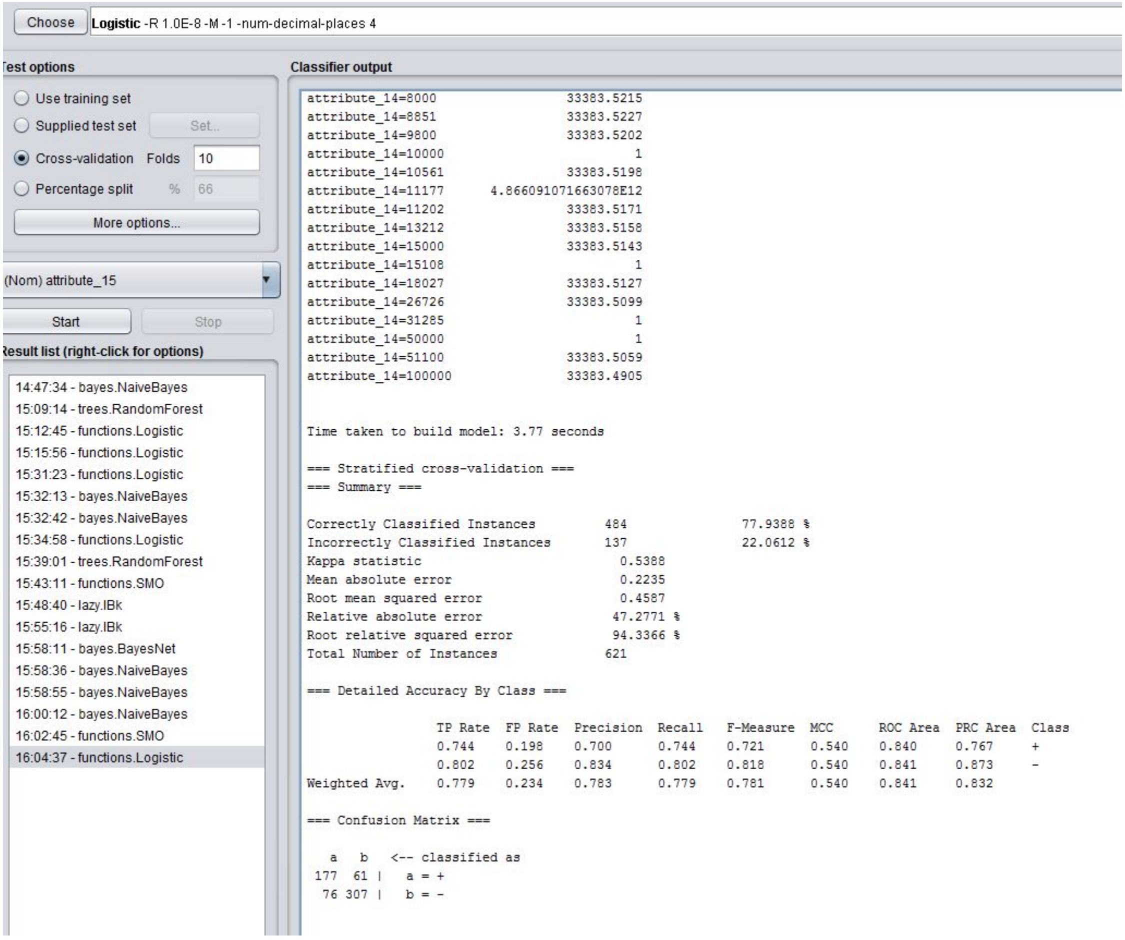
b) Credit Approval Data Set:

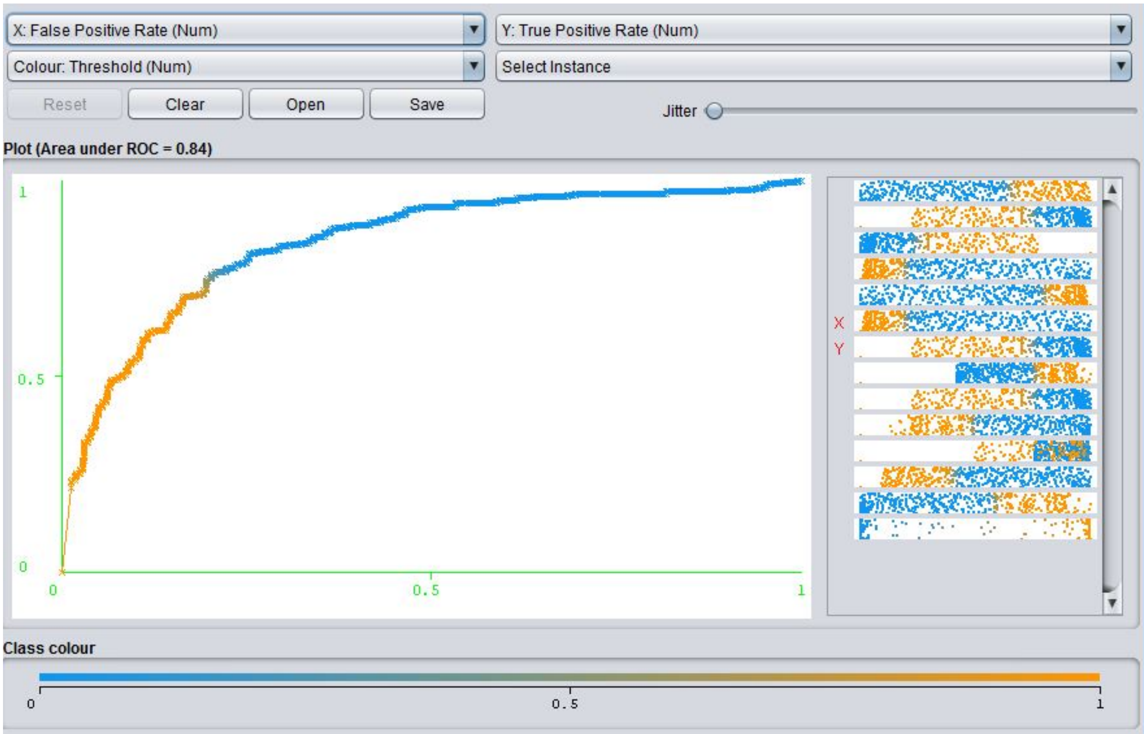
Random Forest:



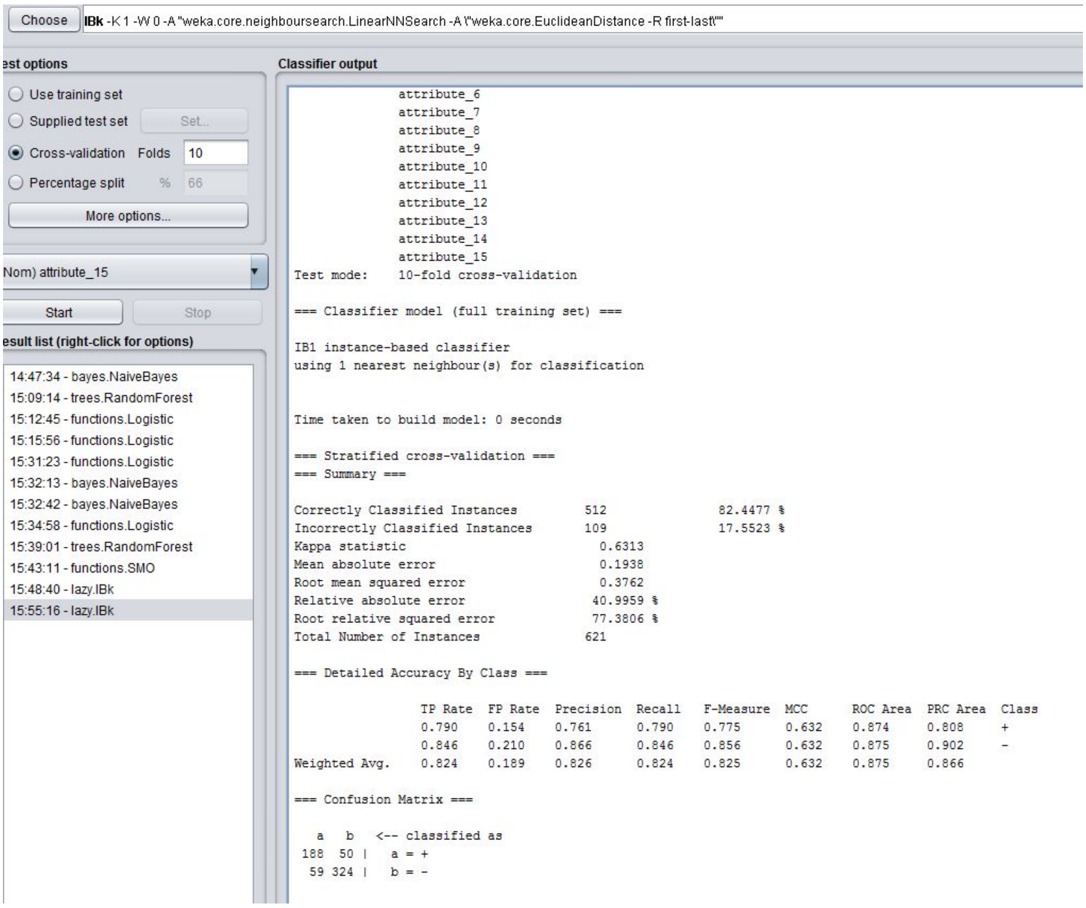


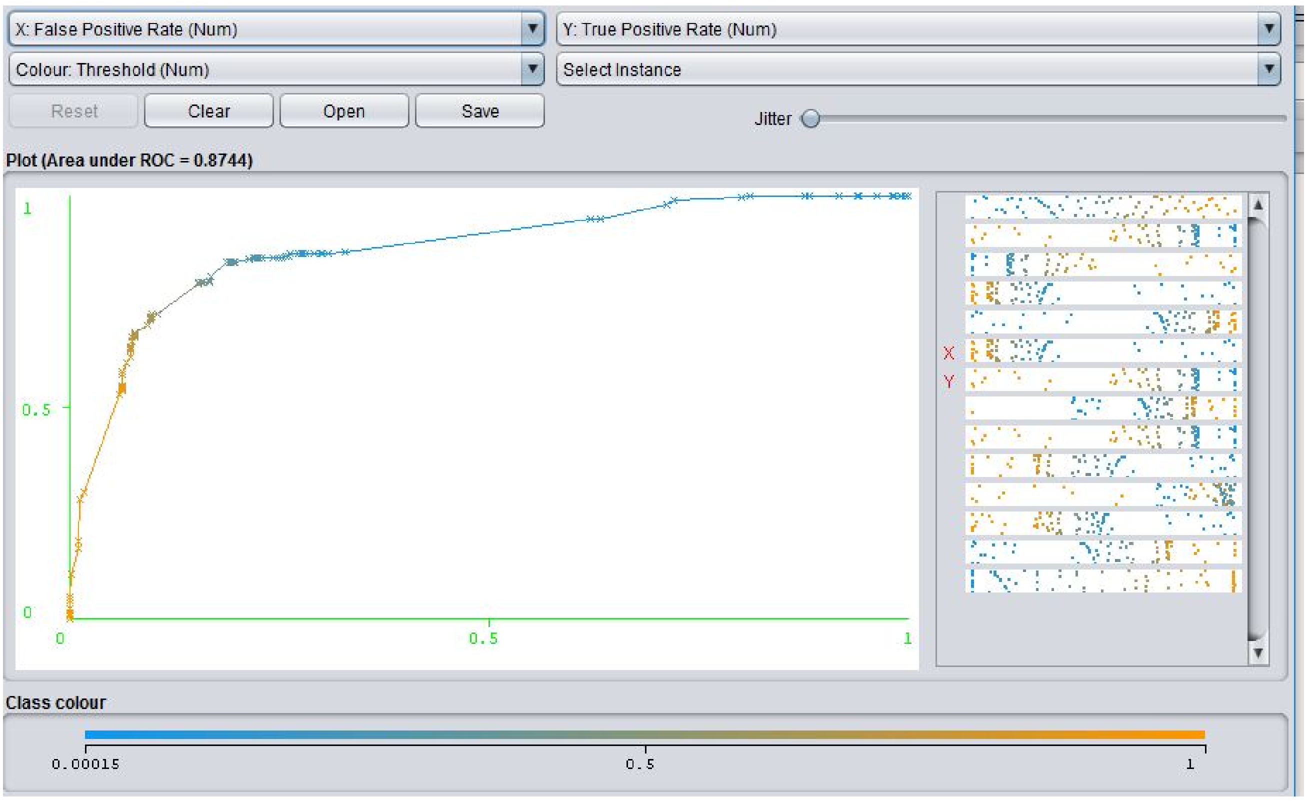
Logistic Regression:



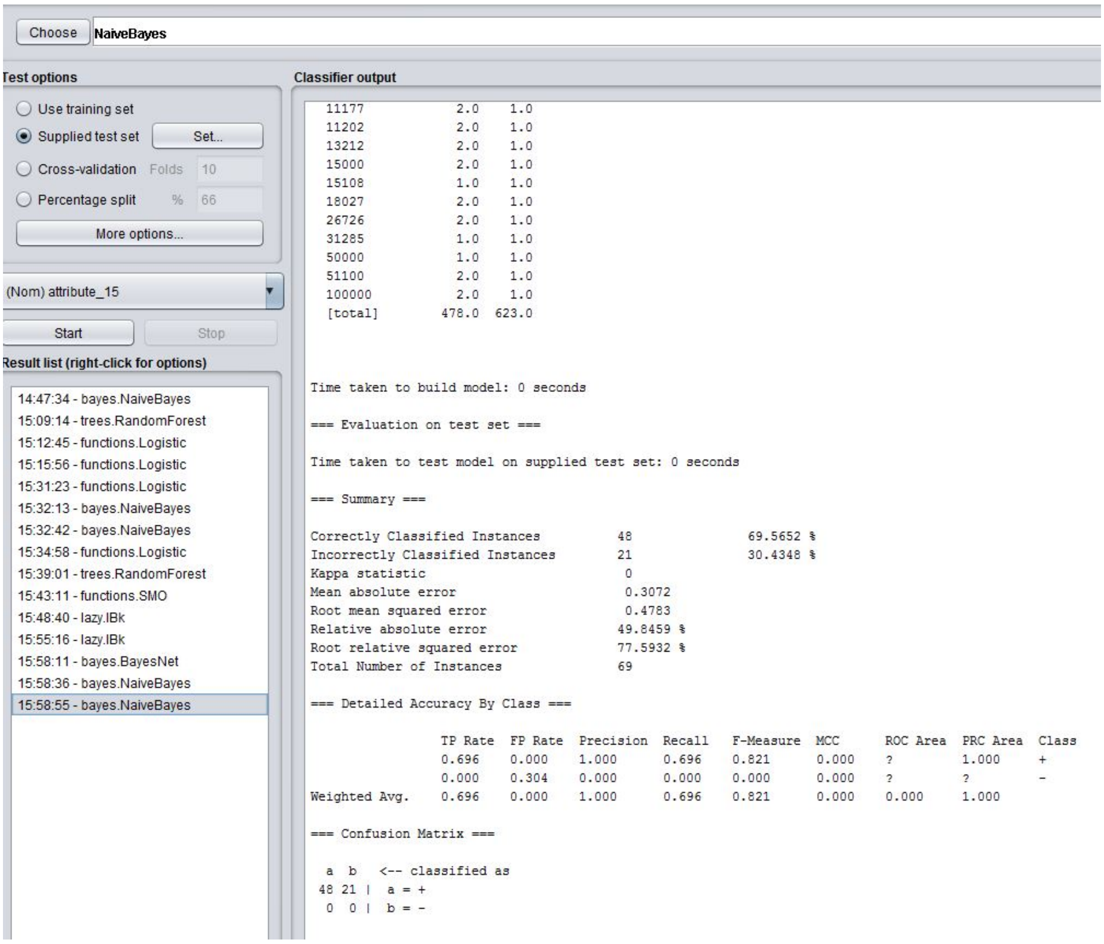


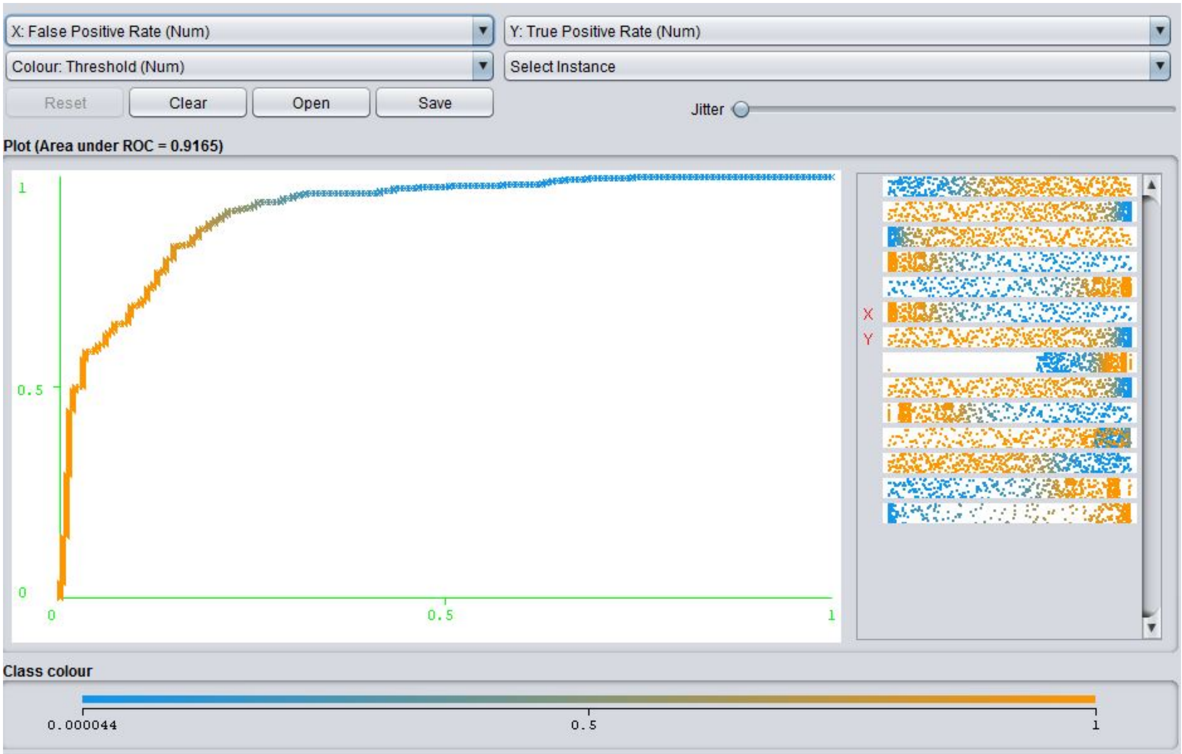
kNN:


Choose
IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
est options
Use training set
Supplied test set
Set...
Cross-validation Folds 10
Percentage split % 66
More options...
Nom) attribute_15
Start
Stop
esult list (right-click for options)
14:47:34 - bayes.NaiveBayes
15:09:14 - trees.RandomForest
15:12:45 - functions.Logistic
15:15:56 - functions.Logistic
15:31:23 - functions.Logistic
15:32:13 - bayes.NaiveBayes
15:32:42 - bayes.NaiveBayes
15:34:58 - functions.Logistic
15:39:01 - trees.RandomForest
15:43:11 - functions.SMO
15:48:40 - lazy.IBk
15:55:16 - lazy.IBk
Classifier output
attribute_6
attribute_7
attribute_8
attribute_9
attribute_10
attribute_11
attribute_12
attribute_13
attribute_14
attribute_15
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
IB1 instance-based classifier
using 1 nearest neighbour(s) for classification
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 512 82.4477 %
Incorrectly Classified Instances 109 17.5523 %
Kappa statistic 0.6313
Mean absolute error 0.1938
Root mean squared error 0.3762
Relative absolute error 40.9959 %
Root relative squared error 77.3806 %
Total Number of Instances 621
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.790 0.154 0.761 0.790 0.775 0.632 0.874 0.808 +
0.846 0.210 0.866 0.846 0.856 0.632 0.875 0.902 -
Weighted Avg. 0.824 0.189 0.826 0.824 0.825 0.632 0.875 0.866
=== Confusion Matrix ===
a b <-- classified as
188 50 | a = +
59 324 | b = -

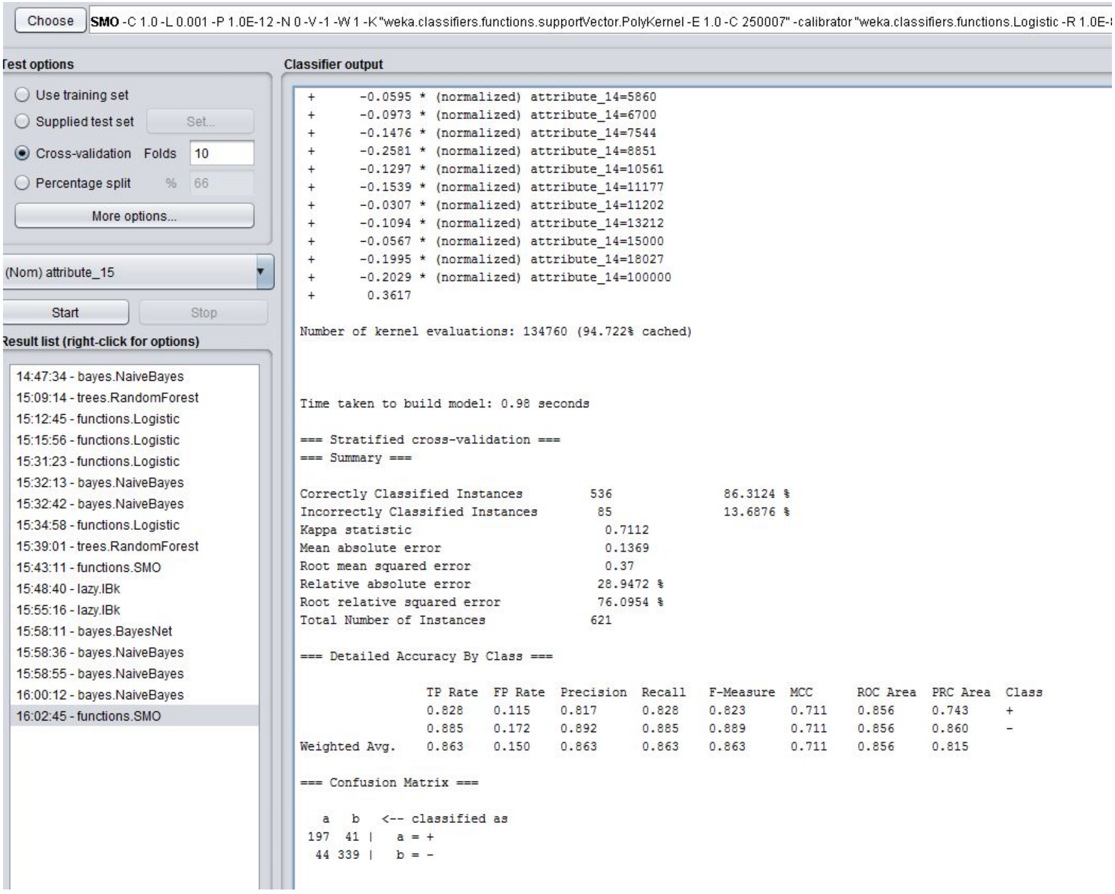

X: False Positive Rate (Num)
Y: True Positive Rate (Num)
Colour: Threshold (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot (Area under ROC = 0.8744)
1
0.5
0
0
0.5
1
X
Y
Class colour
0.00015
0.5
1

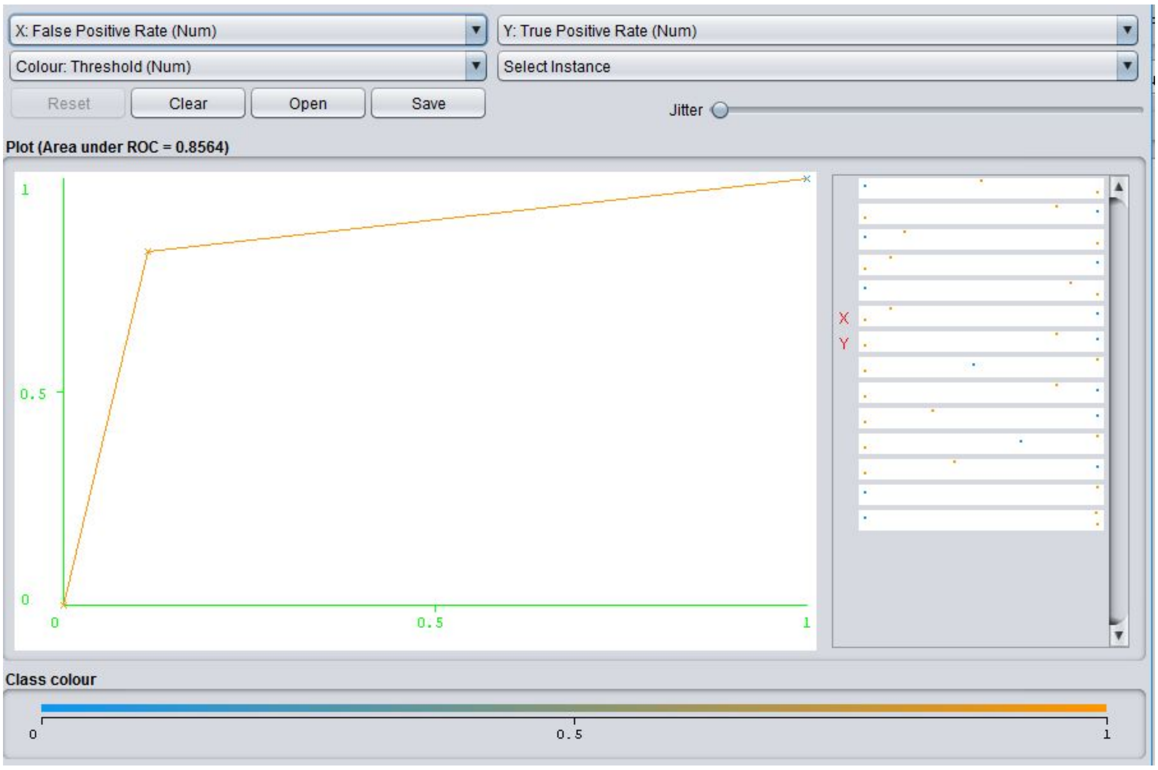
Naïve Bayes:


Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) attribute_15
Start
Stop
Result list (right-click for options)
14:47:34 - bayes.NaiveBayes
15:09:14 - trees.RandomForest
15:12:45 - functions.Logistic
15:15:56 - functions.Logistic
15:31:23 - functions.Logistic
15:32:13 - bayes.NaiveBayes
15:32:42 - bayes.NaiveBayes
15:34:58 - functions.Logistic
15:39:01 - trees.RandomForest
15:43:11 - functions.SMO
15:48:40 - lazy.IBk
15:55:16 - lazy.IBk
15:58:11 - bayes.BayesNet
15:58:36 - bayes.NaiveBayes
15:58:55 - bayes.NaiveBayes
Classifier output
11177 2.0 1.0
11202 2.0 1.0
13212 2.0 1.0
15000 2.0 1.0
15108 1.0 1.0
18027 2.0 1.0
26726 2.0 1.0
31285 1.0 1.0
50000 1.0 1.0
51100 2.0 1.0
100000 2.0 1.0
[total] 478.0 623.0
Time taken to build model: 0 seconds
=== Evaluation on test set ===
Time taken to test model on supplied test set: 0 seconds
=== Summary ===
Correctly Classified Instances 48 69.5652 %
Incorrectly Classified Instances 21 30.4348 %
Kappa statistic 0
Mean absolute error 0.3072
Root mean squared error 0.4783
Relative absolute error 49.8459 %
Root relative squared error 77.5932 %
Total Number of Instances 69
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.696 0.000 1.000 0.696 0.821 0.000 ? 1.000 +
0.000 0.304 0.000 0.000 0.000 0.000 ? ? -
Weighted Avg. 0.696 0.000 1.000 0.696 0.821 0.000 0.000 1.000
=== Confusion Matrix ===
a b <-- classified as
48 21 | a = +
0 0 | b = -


X: False Positive Rate (Num)
Y: True Positive Rate (Num)
Colour: Threshold (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot (Area under ROC = 0.9165)
1
0.5
0
0
0.5
1
X
Y
Class colour
0.000044
0.5
1

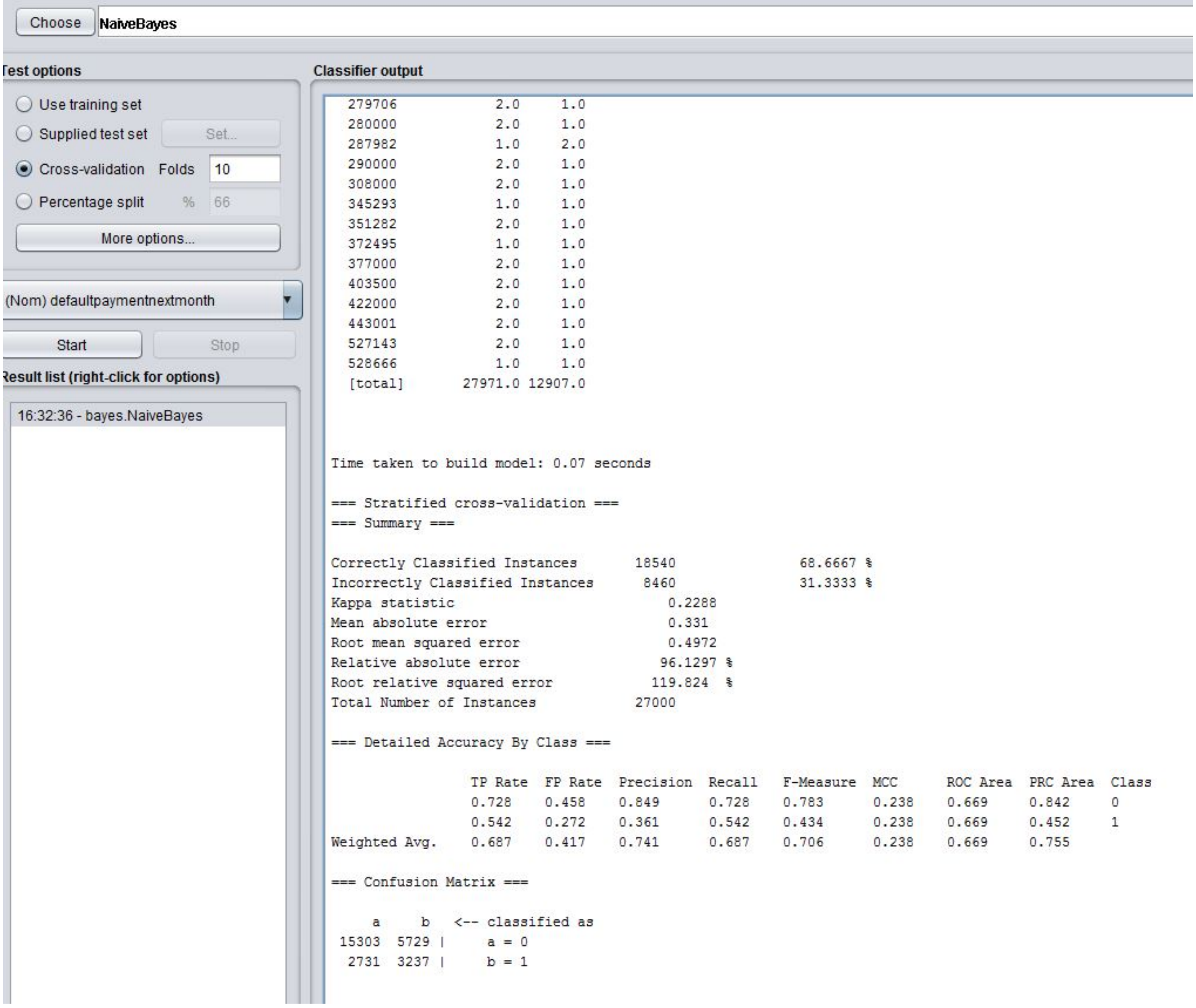
Support Vector Machine (Poly):

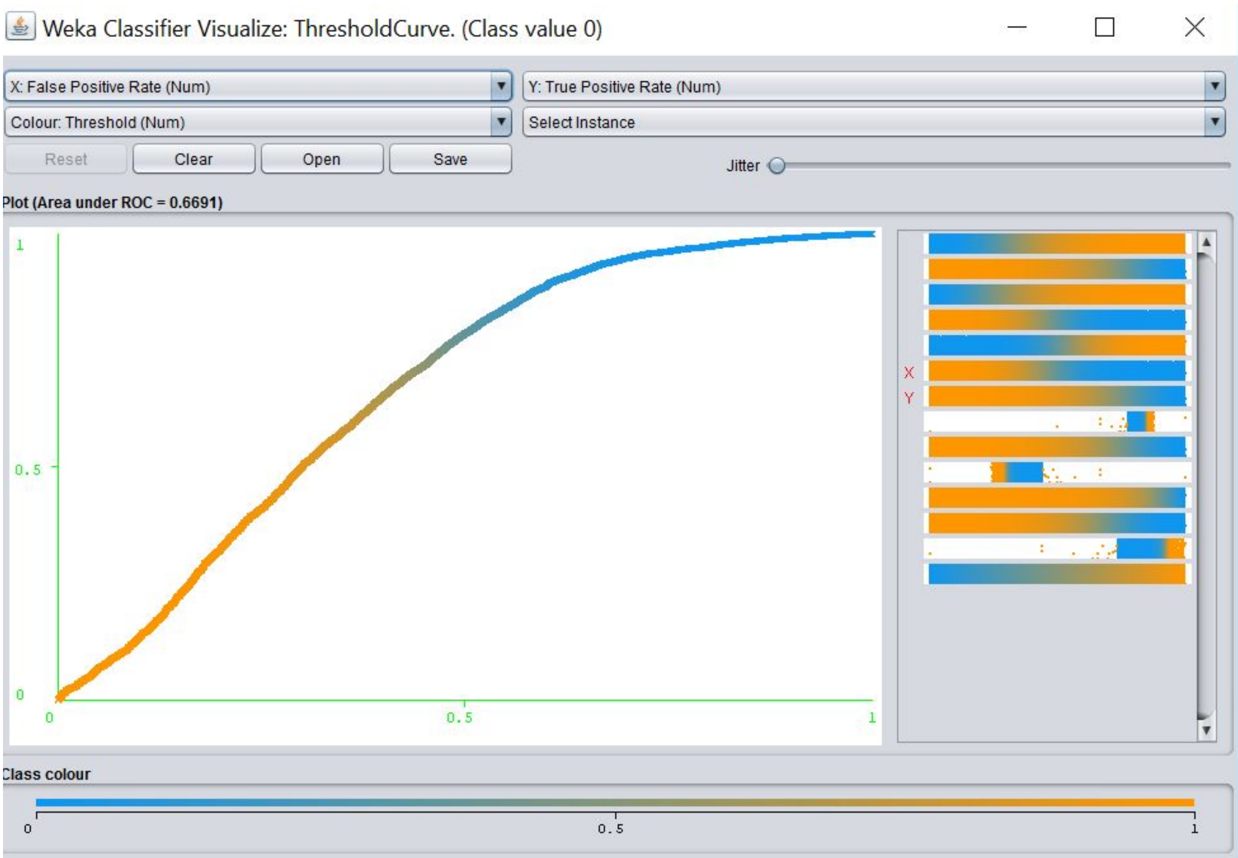




c) Default of Credit Card Clients Data Set:

Naïve Bayes:





ACCURACY SUMMARY:

All the accuracies are based on 10-fold cross validation with 90-10 train-test split.

Model/Accuracies	**Scikit**	**Weka**	**Self**
German:			
Naïve Bayes	72.8%	78%	71.02%
kNN	66.4%	74%	68.95%
SVM	75.2%	75%	-
Logistic Regression	75.7 %	66%	-
Random Forest	70%	74%	-
CRX:			
Naïve Bayes	78%	69.56%	76.71%
kNN	66.8%	82.44%	65.94%
SVM	85.9%	86.43%	-
Logistic Regression	86.9%	77.93%	-
Random Forest	87.4%	73.43%	-
Default of Credit Card:			
Naïve Bayes	37.7%	68.66%	74.03%
kNN	78%	NC	75.5%
SVM	80.3%	NC	-
Logistic Regression	82%	NC	-
Random Forest	81.1%	NC	-

NC: Not computable. Due to the size of the data set, it takes too long to compute and exceeds the heap size as well.

Based on the accuracy table above, we can say that our implementation using Scikit libraries out perform or atleast perform equally well as Weka, but Weka works much better with kNN.

Also, our implementation of kNN and Naïve Bayes performs close to Scikit Libraries.

ANALYSIS:

Naïve Bayes vs kNN vs Logistic Regression vs SVM vs Random Forest

We performed a 10-fold cross validation on each of the data set by splitting them randomly into 90-10 train-test ratio.

1) **Performance on the given datasets:**

a) **German Data Set:**

Accuracy-wise SVM and logistic regression almost perform similar to each other, while Naïve Bayes stands just behind them (for Scikit Learn), whereas Naïve Bayes performs the best on Weka. As per our understanding, this is because this is a linearly separable dataset ie. class label is binary (0 and 1).
Naïve Bayes performs well, since very few features are actually dependent on each other and hence the Naïve Bayes assumption of feature independence benefits the algorithm in this case. Although its performance is brought down a bit due to some continuous values.

But the important point to note here, which was also mentioned in the dataset decription is that for this dataset, it is worse to class a customer as good when they are bad (5), than it is to class a customer as bad when they are good (1).
In other words, the cost of False Positives(FP) is much higher than the cost of False Negatives(FN)
Cost (FP) >>> Cost(FN)

In that sense, Random Forest perfoms the best with the least FPR rate of 2.52% and might be preferred over other higher accuracy algorithms like SVM and Linear Regression, for such a task.

b) **Credit Approval Data Set (CRX):**

Accuracy-wise SVM, Logistic Regression and Random Forest perform pretty well with average accuracy of 85%+. This is again due to its binary nature. Although, we would need a bigger dataset to confirm if all the three of these algorithms can perform well on such a data set.

Naïve Bayes also gives a pretty decent performance of ~77%, because of less no of continuous features and their distinct values. Although discretizing these continuous features could have given a drastic boost to its performance.

Using Scikit and our implementation, kNN doesn't perform well, but performs surprisingly well on Weka.
Since it is again a Credit Approval Dataset, where based on the features we need to determine if a credit card application for a particular person can be approved or not, its cost(FP) >> cost(FN)
In that sense, SVM performs the best and might be the best fit for such kind of datasets.

c) **Default of Credit Card Clients Data Set:**

Again, Accuracy-wise kNN, SVM, Logistic Regression and Random Forest perform pretty well with average accuracy of 80%+. This is again due to its binary nature. Even if not too big, it would be considered to be a decent sized data set, as opposed to the previous ones, and hence kNN takes a lot of time to compute as it needs to find the distance between each test and train points.

Also, this is credit defaulter dataset, where based on the featurs we need predict if a person is going to be a defaulter or not. So in this case, the False Negatives (a person who is going to be a defaulter, but is incorrectly predicted as a non-defaulter) are the most harmful.
Hence, we need to minimize the false negatives for this dataset.

On that basis, kNN and SVM perform the best and are followed by logistic regression.
Even though Random Forest gives highest accuracy, it almost gives thrice or atleast twice as much FN's as them. Hence, we would prefer kNN or SVM over Random Forest for such data set.

Sci-kit based Naïve Bayes gives a low performance of ~37%, probably because of a very high number of no of continuous features and their distinct values, but our implemented Naïve Bayes performs pretty well with an accuracy of 74%. We almost thought there is some bug in our code, but then felt assured by testing the same on Weka on getting an accuracy of ~69%
Also, discretizing these continuous features could have given a pretty good boost to its performance. Our implementation of kNN performs almost on par with the Scikit based kNN.
Also, we couldn't finish the testing of this dataset on Weka due to its size, which lead to HeapOutOfMemoryException for all the algorithms expect Naïve Bayes.

2) Miscellaneous Observations:

Overall, Logistic and SVM have better area under ROC curve and higher accuracy.
Others perform similar way but the area under ROC curve aren't as good as these algorithm.

Class label is binary (1 and 2) so all the classifiers perform good on the dataset.
All three algorithms have an average accuracy of 80%+ for the datasets.

All the algorithms except kNN have more than 80% accuracy. Even the area under ROC curve is more or less than the same. kNN doesn't work better may be because of the k may not be optimal.

The major difference in the third dataset's accuracy and area under ROC curve may be due to dataset size difference.

Our implementations for all dataset performs at par with scikit learn libraries.

REFERENCES:

http://scikit-learn.org/stable/
https://archive.ics.uci.edu/ml/datasets.html
http://www.cs.waikato.ac.nz/ml/weka/documentation.html