PATTERN RECOGNITION

Assignment 2

Name: SowndaryaKrishnamoorthy

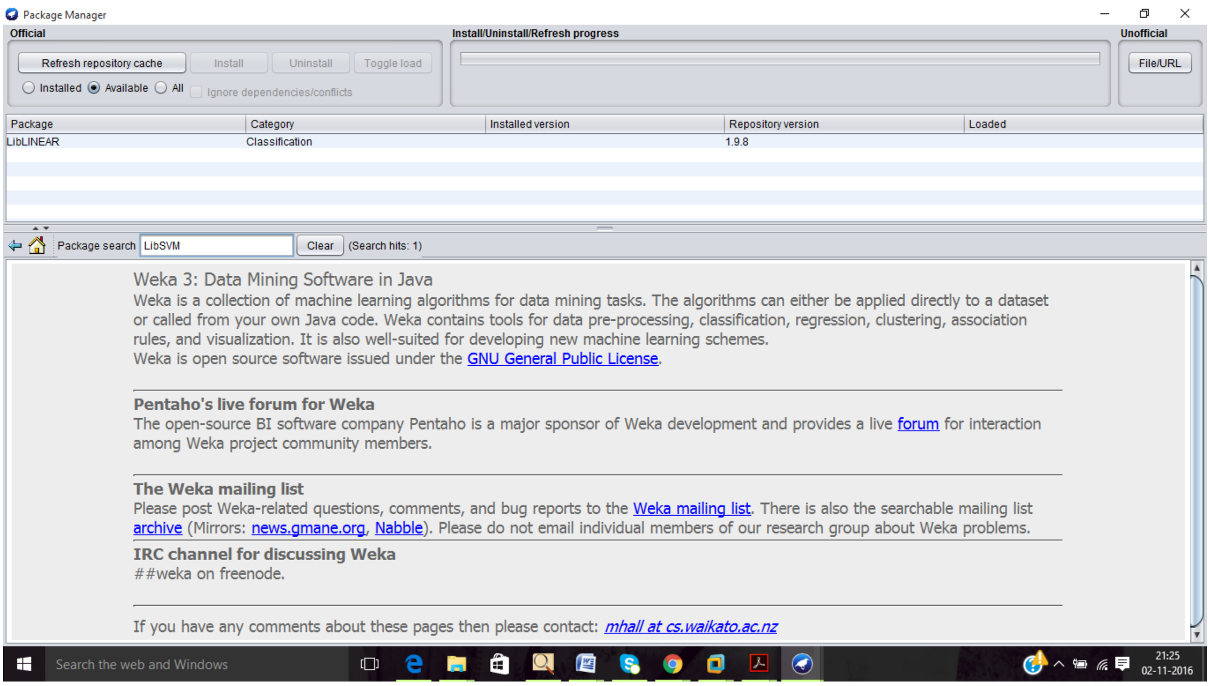
Student ID: 104654482

1) **INSTALLING LibSVM:**
   Support Vector Machines (SVM) is used to obtain a linear classifier based on convex optimization problem and non-linear classifier using kernel trick.

   LibSVM is an integrated software to support SVM multi-class classification by providing interface for users.

   Install LibSVM package in Weka to carry out SVM classification.

   Screenshot: LibSVM installed in Weka



2) **SVM classifiers with different kernels**

The main idea of SVM is to obtain a linear or non-linear classifier based on the convex optimization problem.

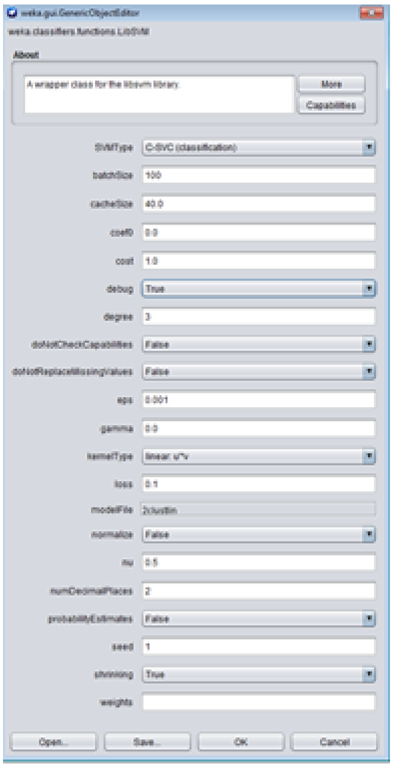
The choice of the kernel depends on the problem domain. The three kernels discussed are linear, polynomial and radial basis function (RBF).

i) CLUSTER IN CLUSTER DATASET:-

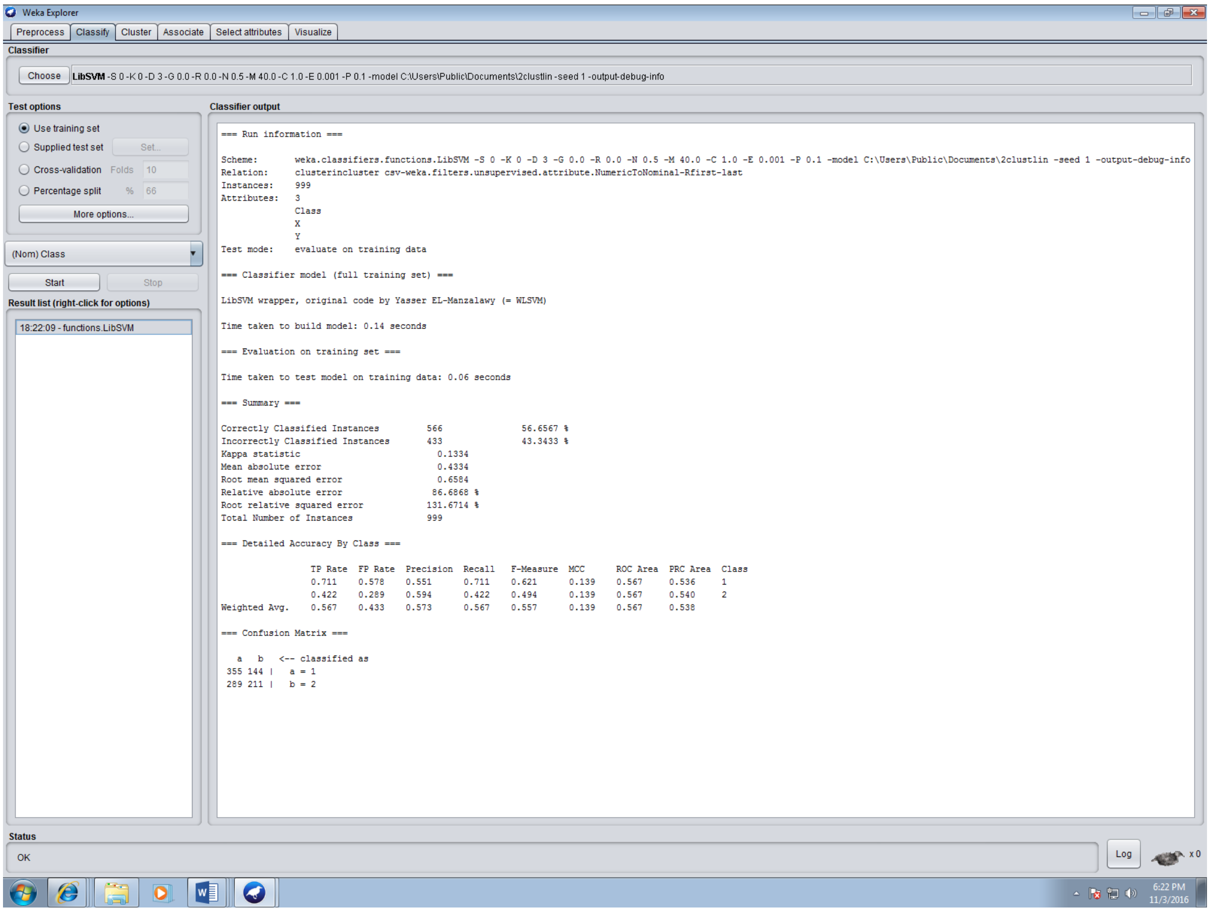
SVM- L (Linear kernel)

The linear kernel has a separating hyper plane satisfies $z_k g(y_k) >= 1$ for k=1, .., n. The margin is any positive distance from the hyper plane to the samples and the aim of SVM is to find the separating with the largest margin to classify better.

Screenshot: Choosing the Kernel Type as Linear and changing parameters after selecting LibSVM Classifier



Screenshot: Efficiency measures for Linear Classifier



SVM –P (Polynomial kernel)

The kernel $K(x,x')=(x\cdot x')^d$ gives the same result as the explicit mapping + dot product :

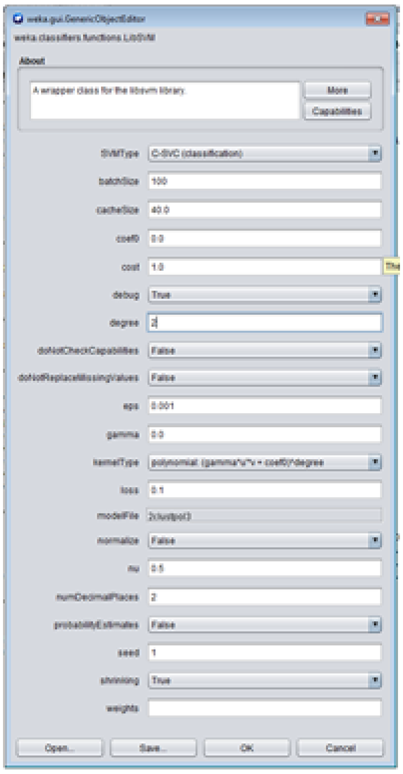
Φ: R2→R3 => $(x_1, x_2) \rightarrow (z_1, z_2, z_3)$

The phi (Φ) function denotes arbitrary values of x. The kernel is the inner product of two vectors.

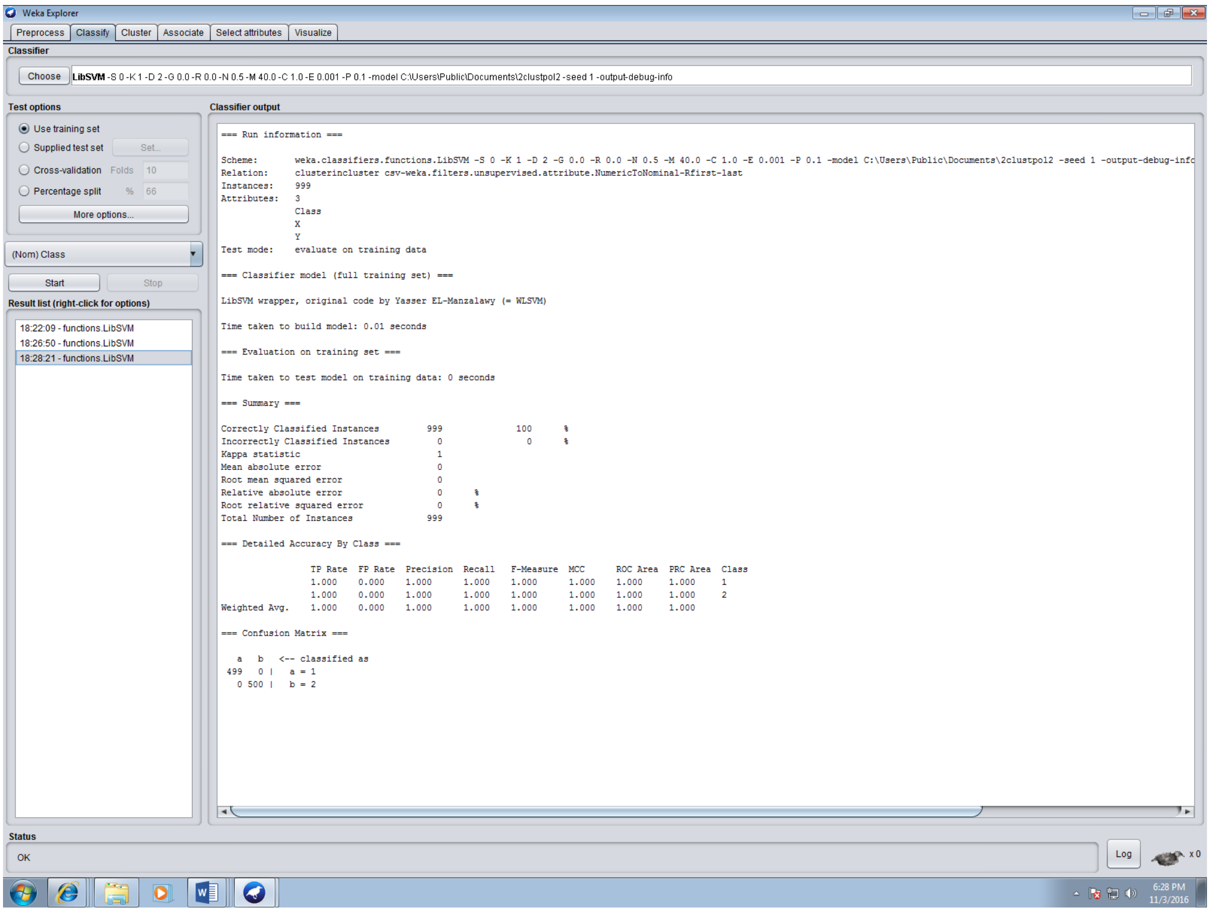
The degree for the polynomial kernel ranges from **2 to 8.** Three different degrees of polynomial are discussed below:

Degree=2:

Screenshot: Changing Parameters for SVM-P by varying degree

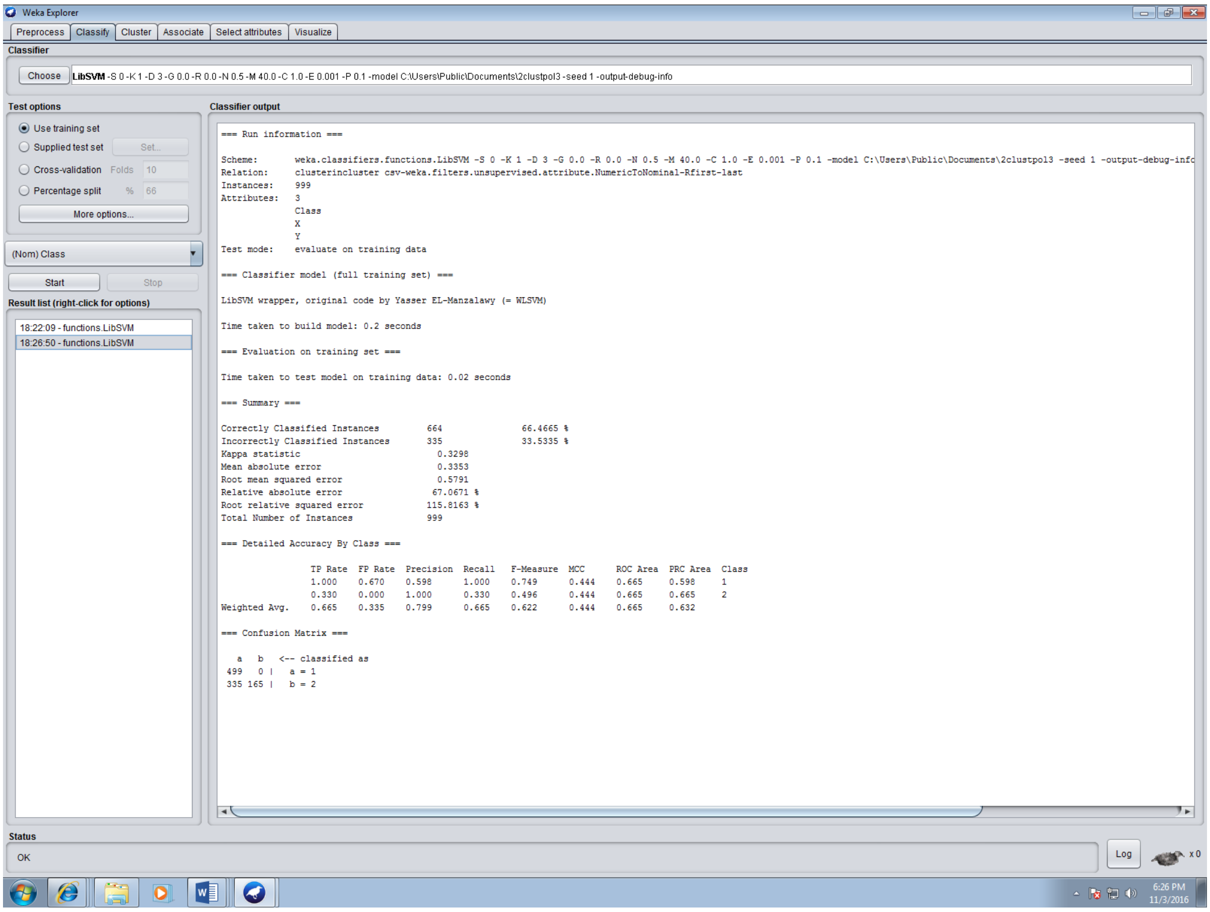


Screenshot: Efficiency measures for Polynomial Classifier with degree 2



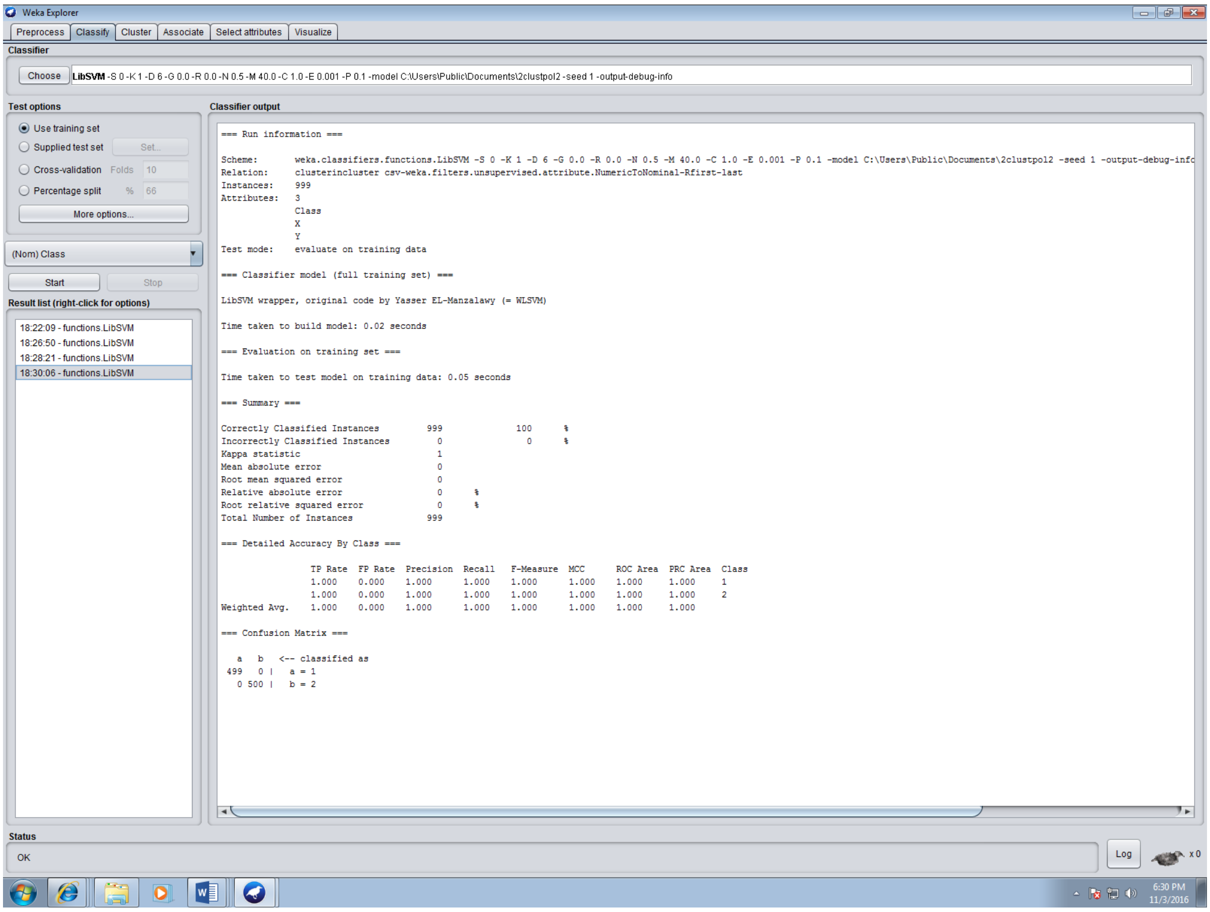
Degree=3:

Screenshot: Efficiency measures for Polynomial Classifier with degree 3



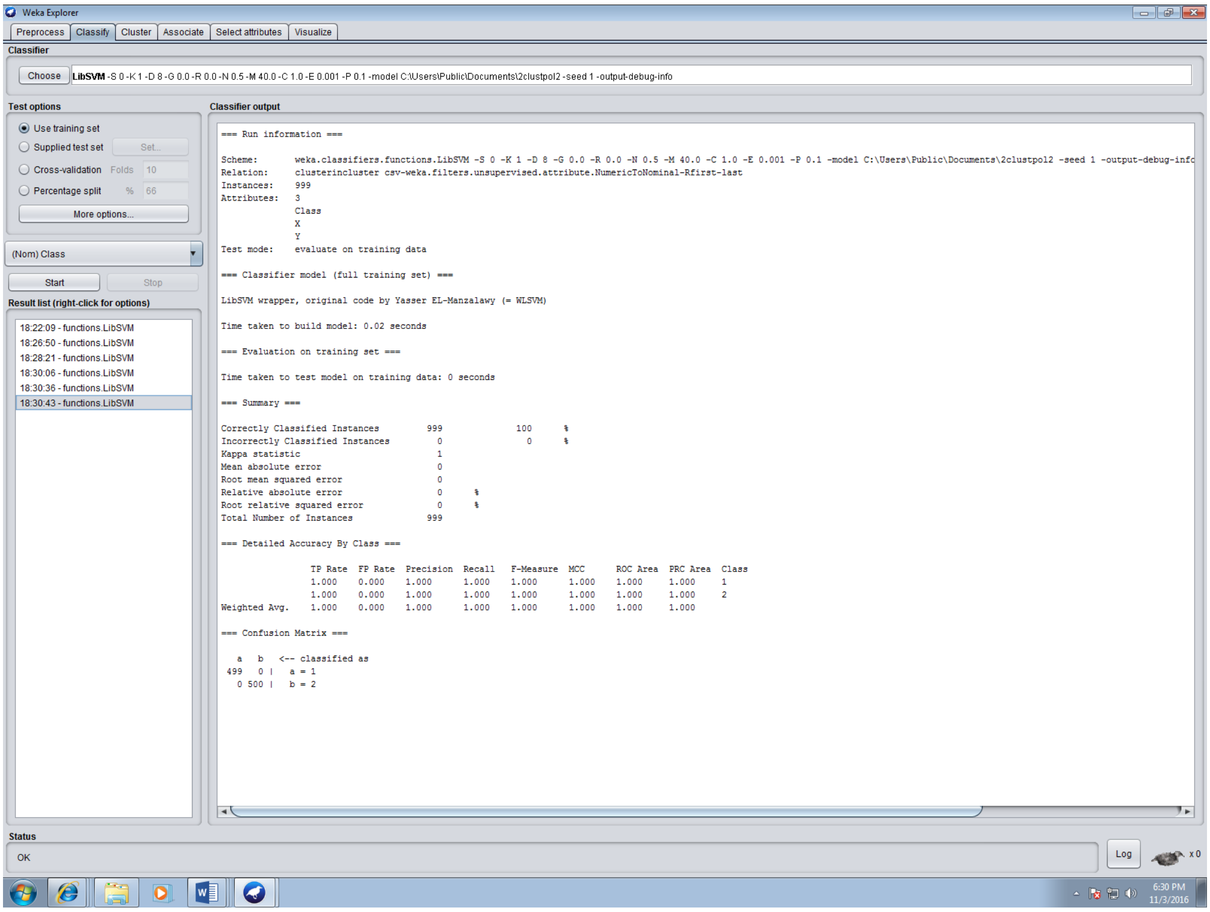
Degree=6:

Screenshot: Efficiency measures for Polynomial Classifier with degree 6



Degree=8:

<u>Screenshot: Efficiency measures for Polynomial Classifier with degree 8</u>



LibSVM-RBF

The RBF kernel K (x, x') =exp $(-\gamma||x_i - x||^2)$ is one of the most popular kernel functions. It adds a 'bump' around each data point.

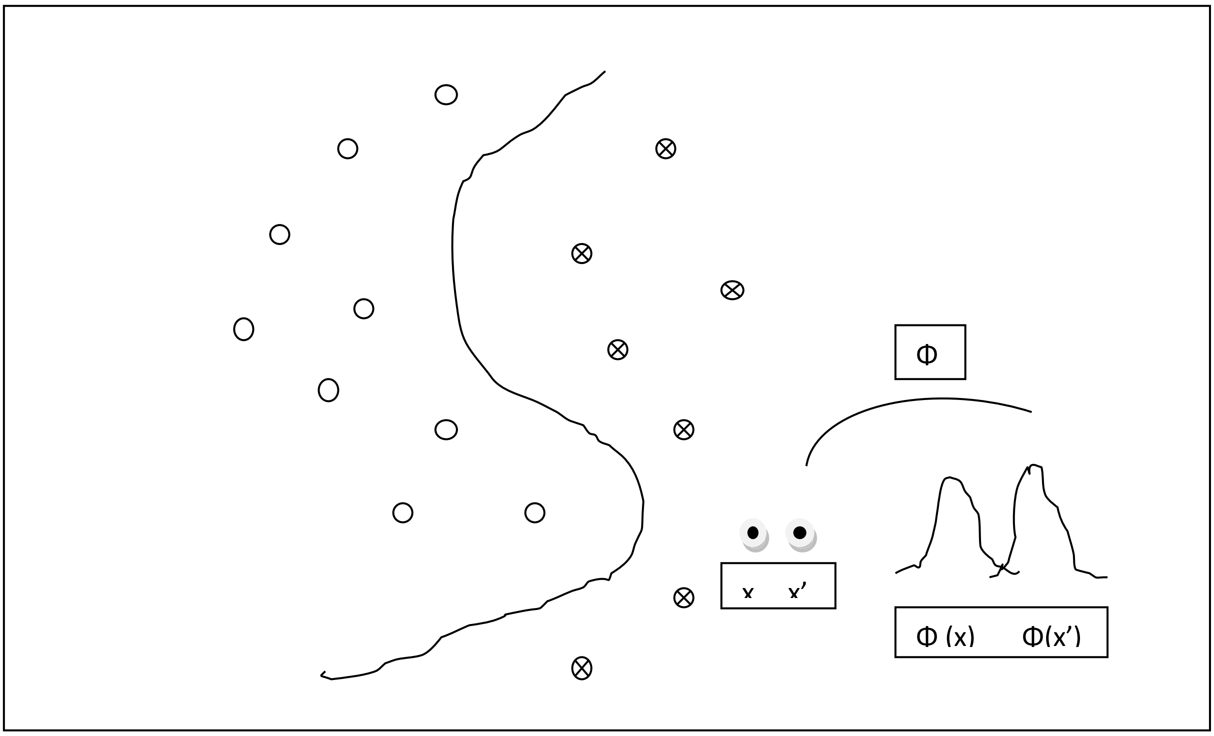
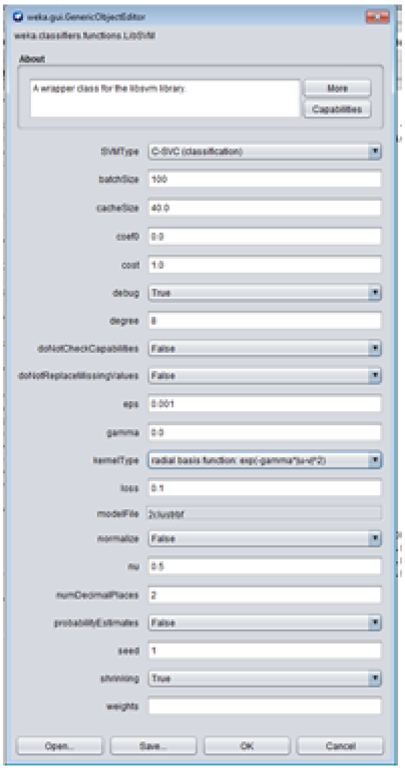


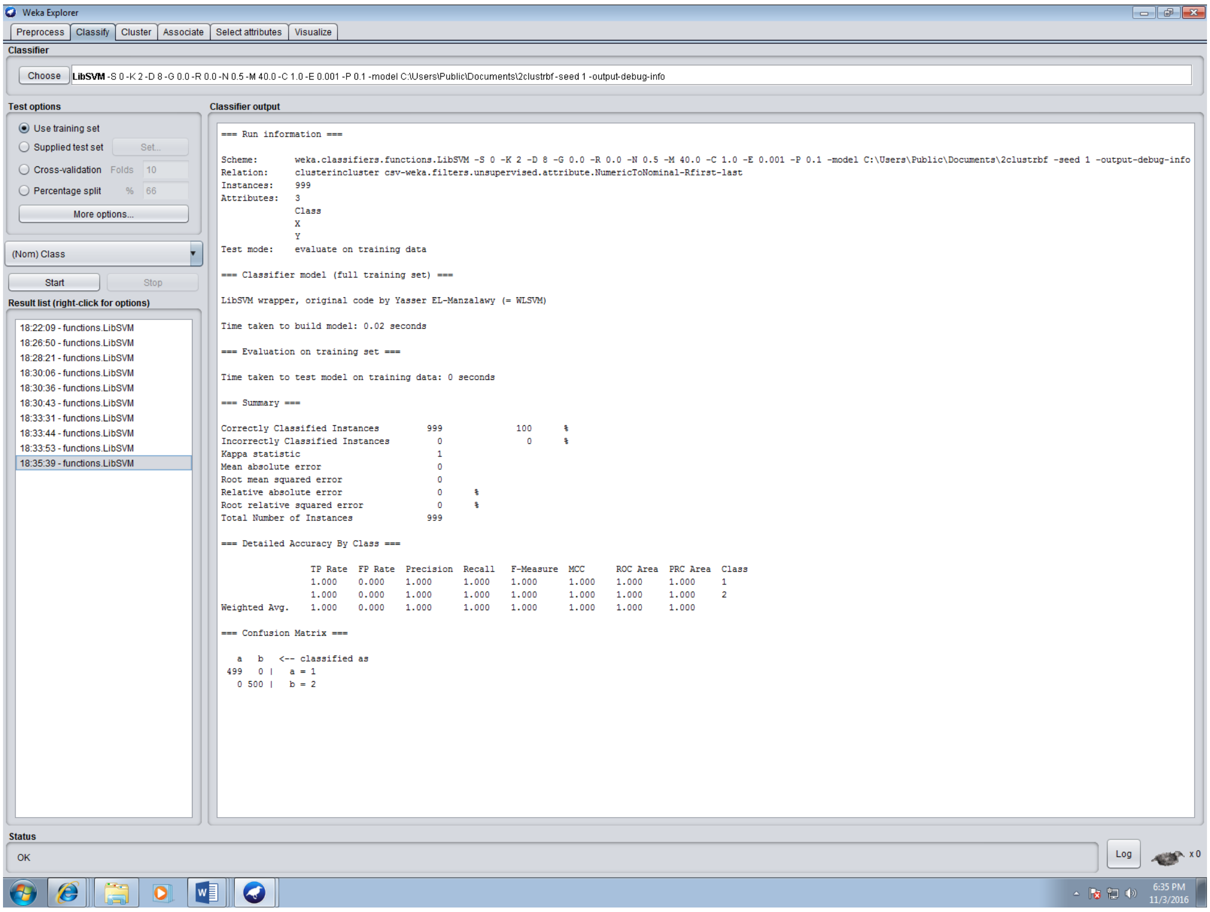
Figure: Radial Basis Function Kernel used to classify class samples

It is also known as Gaussian kernel where gamma controls the shape of the separating function.

Screenshot: Choosing the Kernel Type as Radial Basis Function (RBF) and changing paramters after selecting LibSVMClassifer



Screenshot: Efficiency measures for Radial Basis Classifier (RBF) Classifier

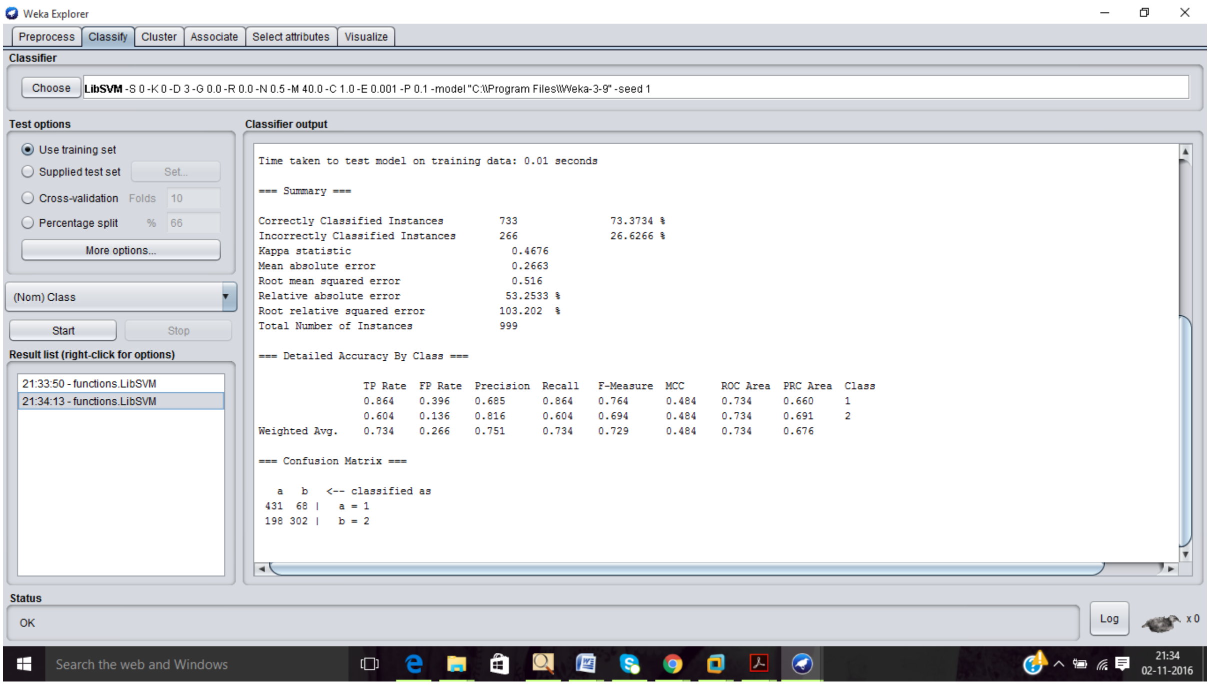


ii) HALF KERNEL DATASET

SVM- L(Linear kernel)

Screenshot: Choosing the Kernel Type as Linear and changing paramters after selecting LibSVMClassifer

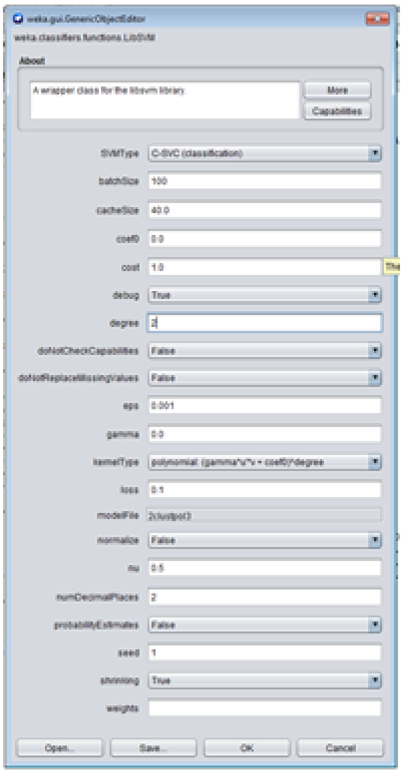
Screenshot: Efficiency measures for Linear Classifier



SVM –P (Polynomial kernel)

Degree=2:

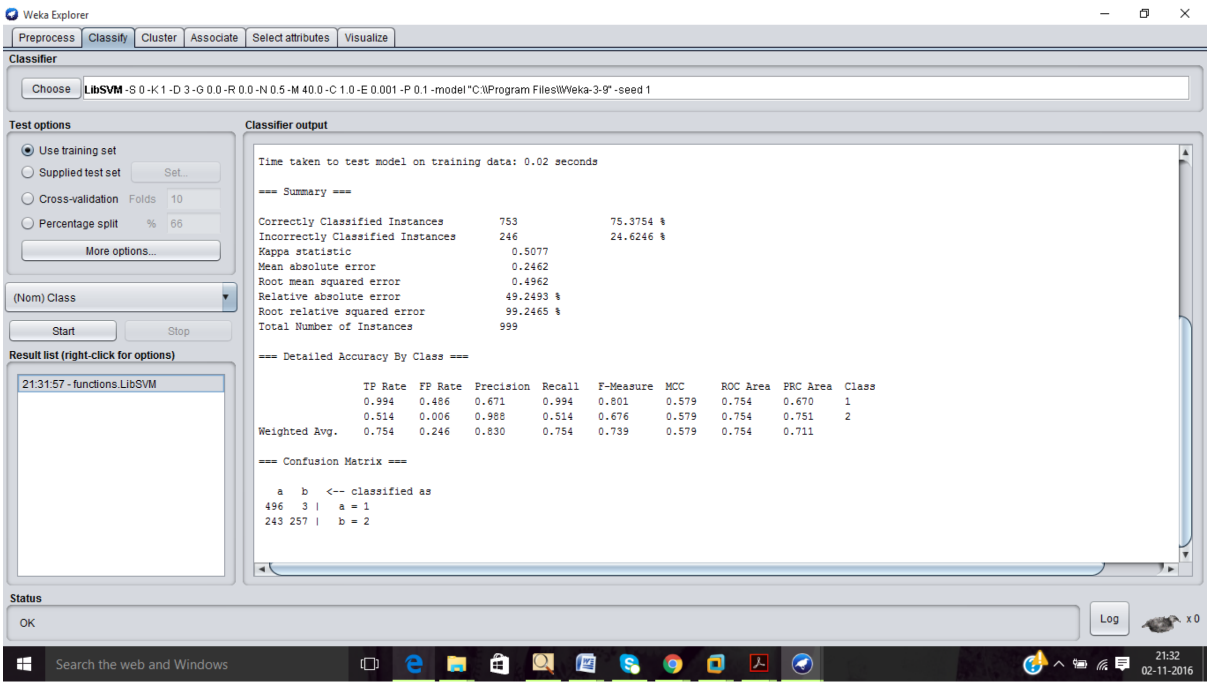
Screenshot: Changing Parameters for SVM-P by varying degree



Screenshot: Efficiency measures for Polynomial Classifier with degree 2

Degree=3:

Screenshot: Efficiency measures for Polynomial Classifier with degree 3



Degree=6:

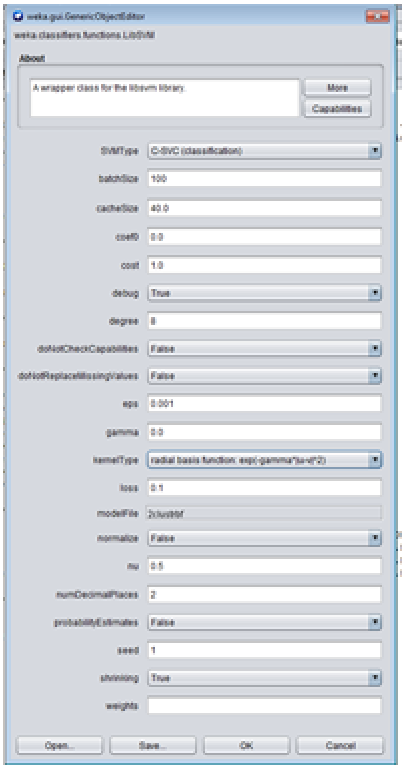
Screenshot: Efficiency measures for Polynomial Classifier with degree 6

Degree=8:

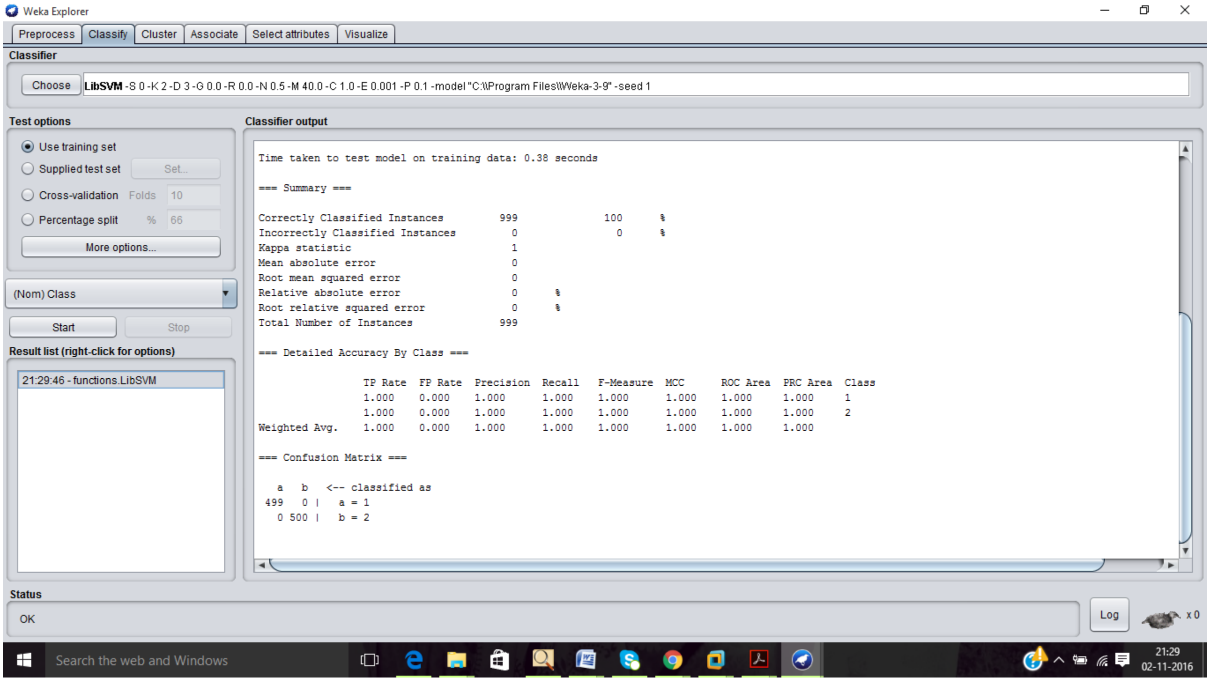
Screenshot: Efficiency measures for Polynomial Classifier with degree 8

LibSVM-RBF

Screenshot: Choosing the Kernel Type as Radial Basis Function (RBF) and changing paramters after selecting LibSVMClassifer



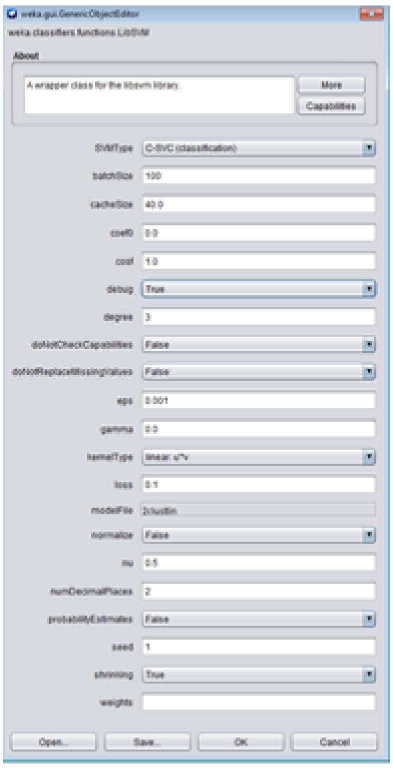
Screenshot: Efficiency measures for Radial Basis Classifier (RBF) Classifier



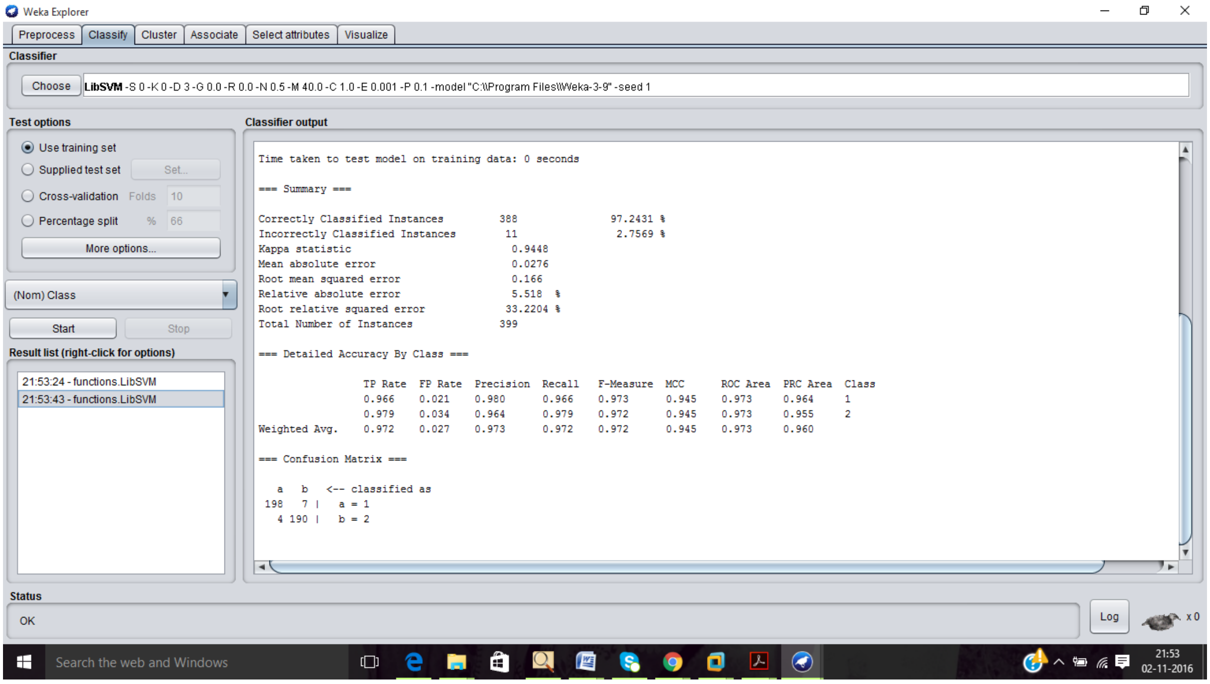
iii) TWO GAUSSIANS DATASET

<u>SVM- L(Linear kernel)</u>

<u>Screenshot: Choosing the Kernel Type as Linear and changing paramters after selecting LibSVMClassifer</u>



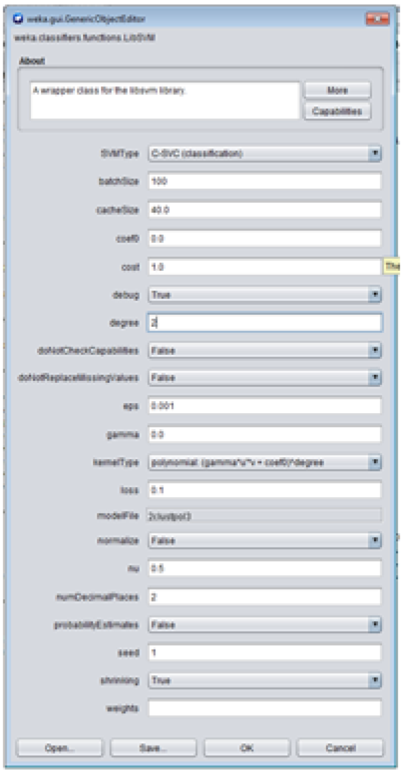
Screenshot: Efficiency measures for Linear Classifier



SVM –P (Polynomial kernel)

Degree=2:

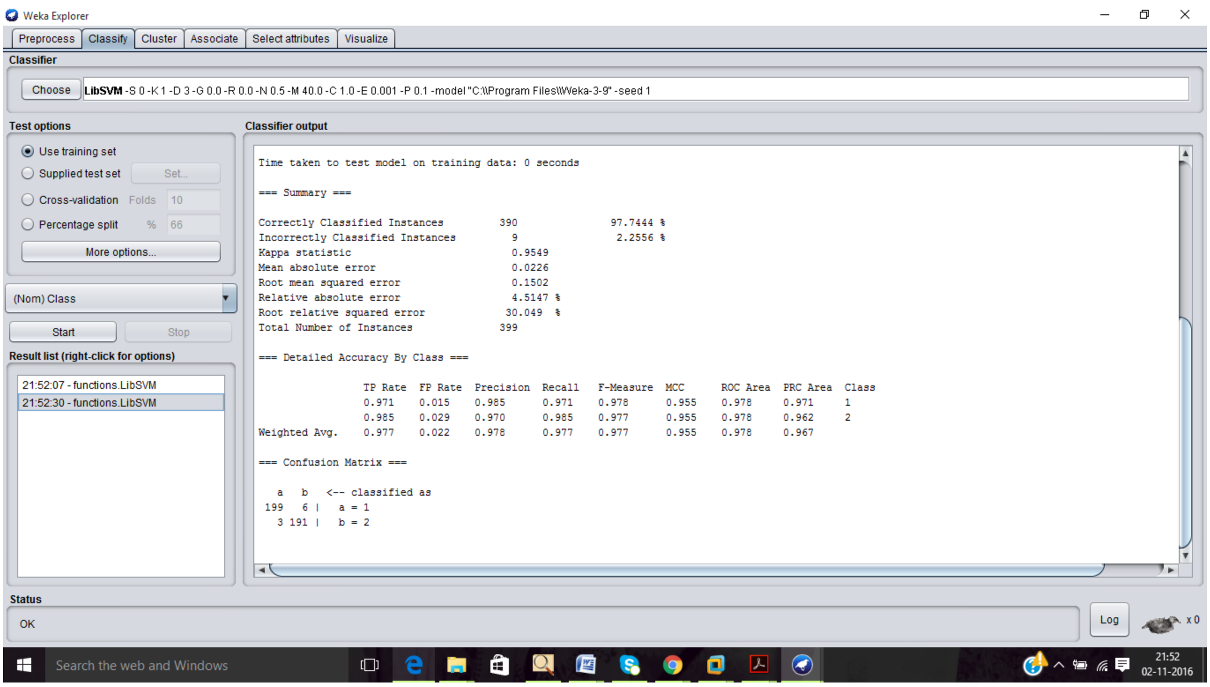
Screenshot: Changing Parameters for SVM-P by varying degree



Screenshot: Efficiency measures for Polynomial Classifier with degree 2

Degree=3:

Screenshot: Efficiency measures for Polynomial Classifier with degree 3



Degree=6:

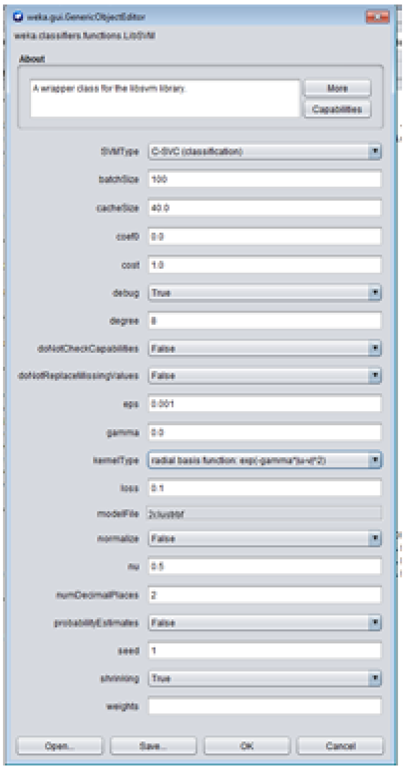
Screenshot: Efficiency measures for Polynomial Classifier with degree 6

Degree=8:

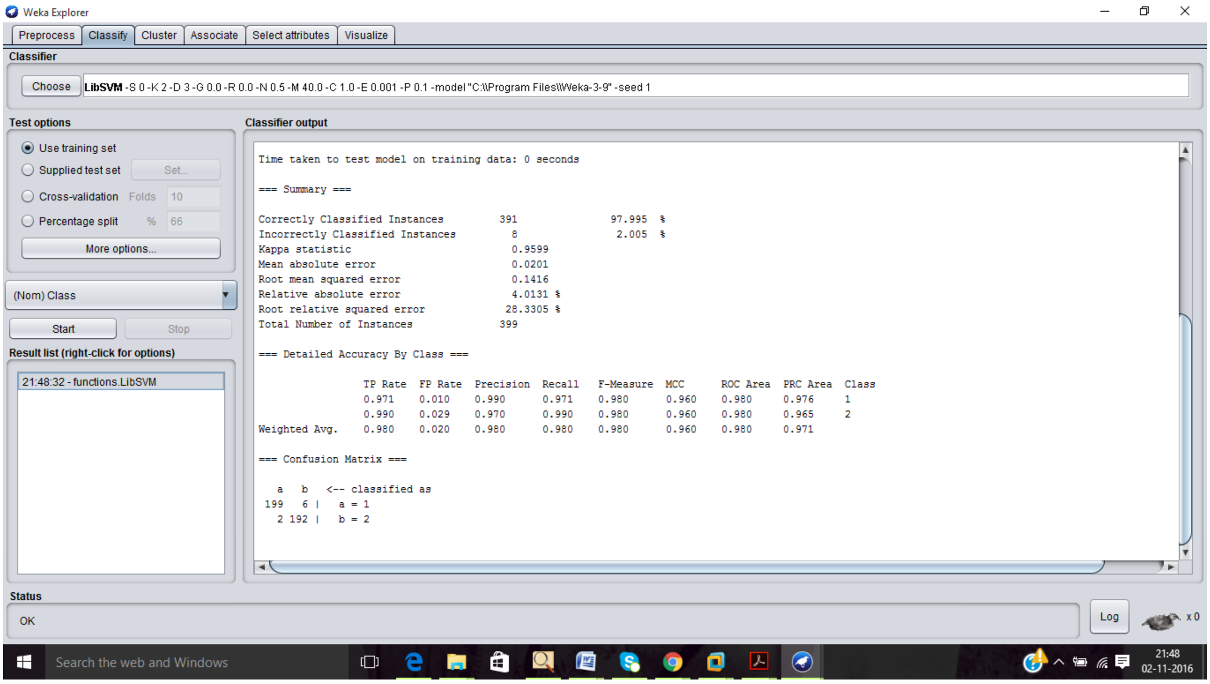
Screenshot: Efficiency measures for Polynomial Classifier with degree 8

LibSVM-RBF

Screenshot: Choosing the Kernel Type as Radial Basis Function (RBF) and changing paramters after selecting LibSVMClassifer



Screenshot: Efficiency measures for Radial Basis Classifier (RBF) Classifier

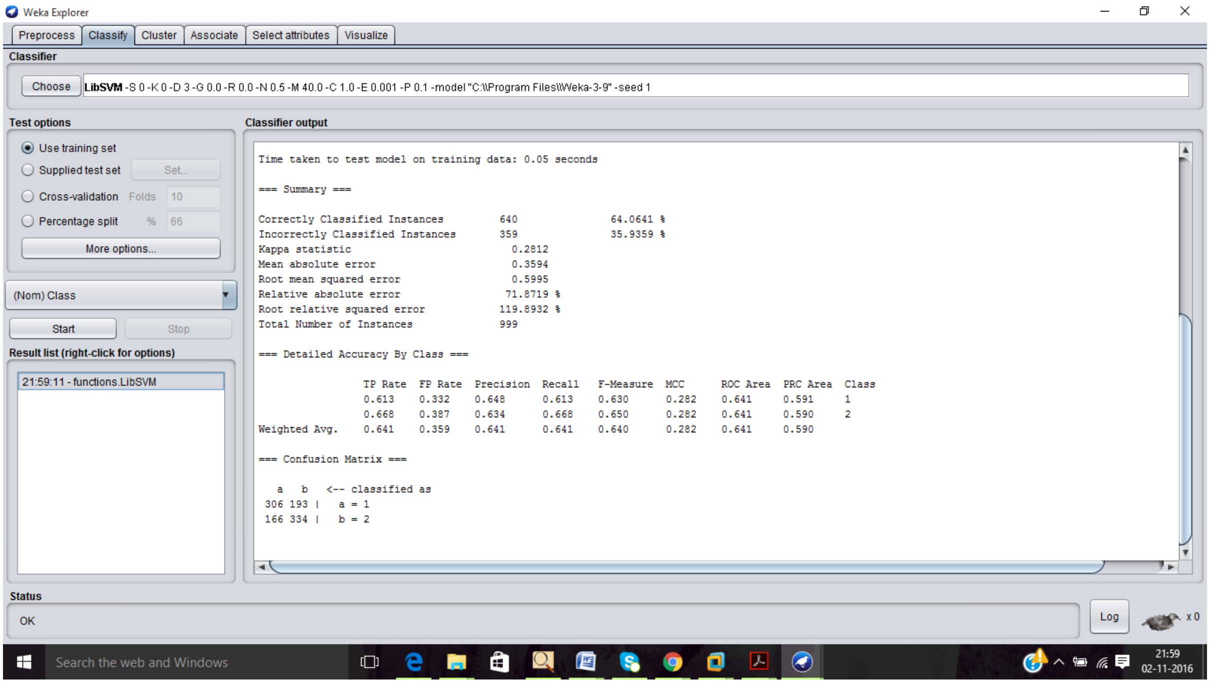


iv) TWO SPIRALS DATASET:

SVM- L(Linear kernel)

Screenshot: Choosing the Kernel Type as Linear and changing parameters after selecting LibSVM Classifier

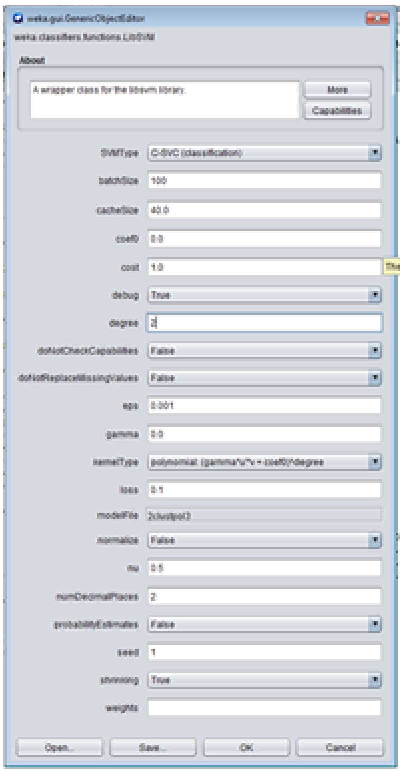
Screenshot: Efficiency measures for Linear Classifier



<u>SVM –P (Polynomial kernel)</u>

Degree=2:

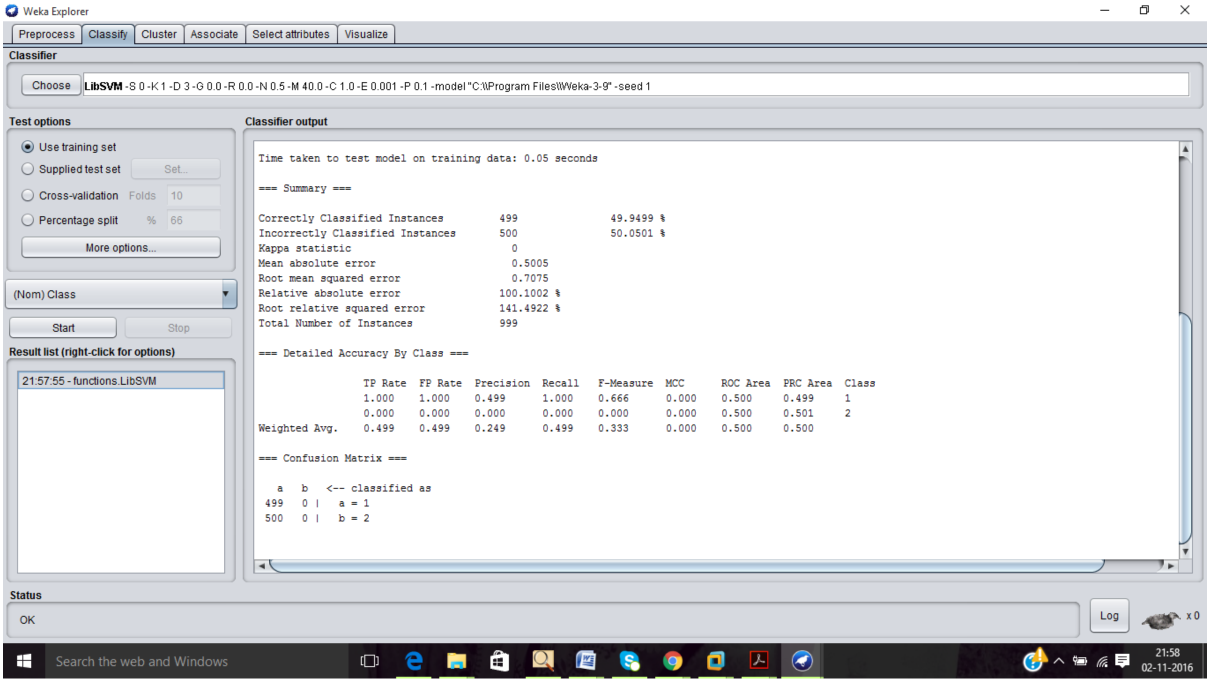
<u>Screenshot: Changing Parameters for SVM-P by varying degree</u>



<u>Screenshot: Efficiency measures for Polynomial Classifier with degree 2</u>

Degree=3:

Screenshot: Efficiency measures for Polynomial Classifier with degree 3



Degree=6:

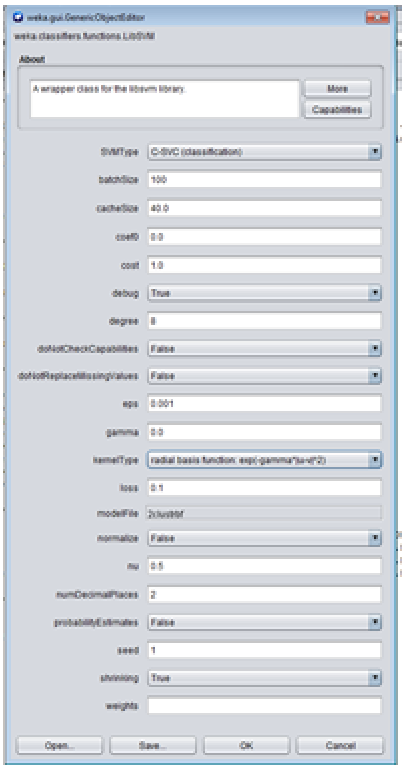
Screenshot: Efficiency measures for Polynomial Classifier with degree 6

Degree=8:

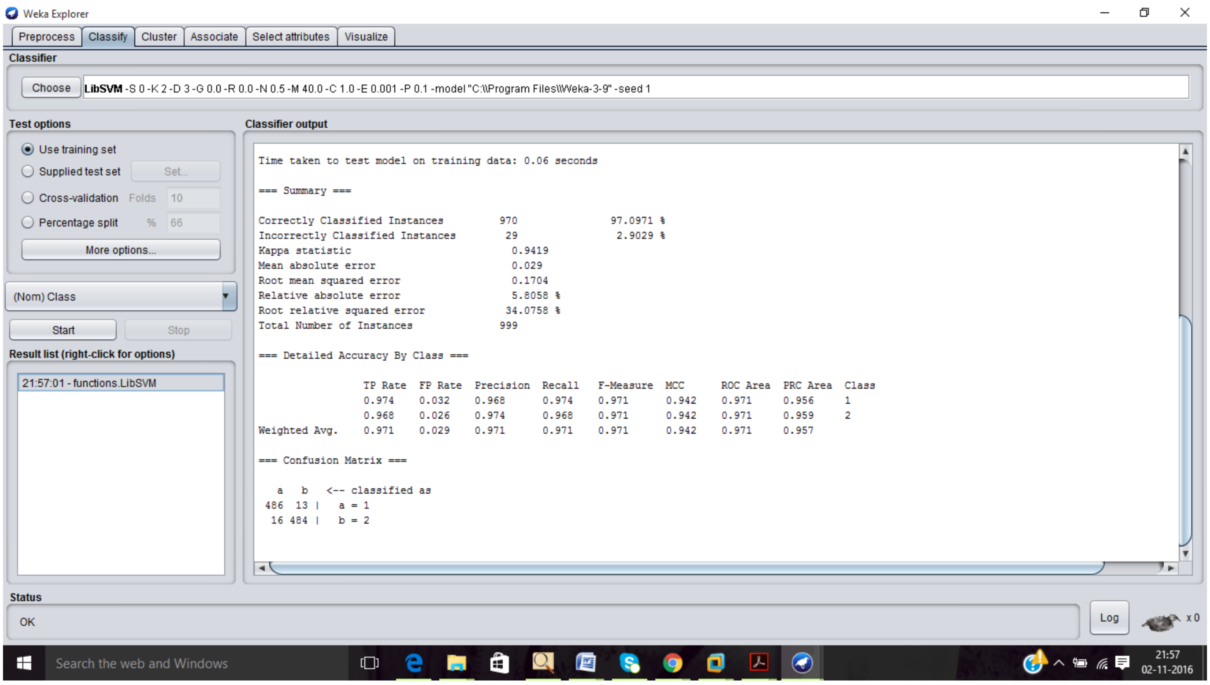
Screenshot: Efficiency measures for Polynomial Classifier with degree 8

LibSVM-RBF

Screenshot: Choosing the Kernel Type as Radial Basis Function (RBF) and changing paramters after selecting LibSVMClassifer



Screenshot: Efficiency measures for Radial Basis Classifier (RBF) Classifier



3) **THREE CLASSIFIERS FOR DEFAULT PARAMETERS USING 10 FOLD CROSS VALIDATION**

The classifiers have been ran with default parameters on 4 datasets using 10 fold cross validation, where class 1 corresponds to positive and class 2 corresponds to negative.

CROSS VALIDATION-

In cross validation, datasets are randomly split into training dataset and validation dataset (testing dataset). To reduce the generalisation error, the classifier is trained until it reaches a minimum of this validation error.

Given a dataset, D=D1 U D2 U D3 ... U Dc and Di is split into two subsets Di1 and Di2.

Then, training dataset is D11 U D21 ... U Dc1

Testing dataset is D12 U D22 U ... Dc2

A generalisation of this method is the M-FOLD CROSS VALIDATION. The datasets are divided into m disjoint sets of equal size and trained m times. Every time validation happens, one different set in used for testing and remaining samples are used for training. We can repeat the process, by swapping the training and testing sets. The estimated performance is the mean of these m errors.

10 FOLD CROSS VALIDATION-

The first 90% of the patterns is used as a standard training set for setting free parameters in the classifier model and the other 10% is the validation set and is meant to represent the full generalization task. In Weka, the m value can be adjusted to perform validation. Counters are used to run for 10 iterations and the total number of TN (True Negatives), TP (True Positives), FN (False Negatives) and FP (False Positives). The total number of samples n is given by the summation of all the parameters.

PERFORMANCE MEASURES

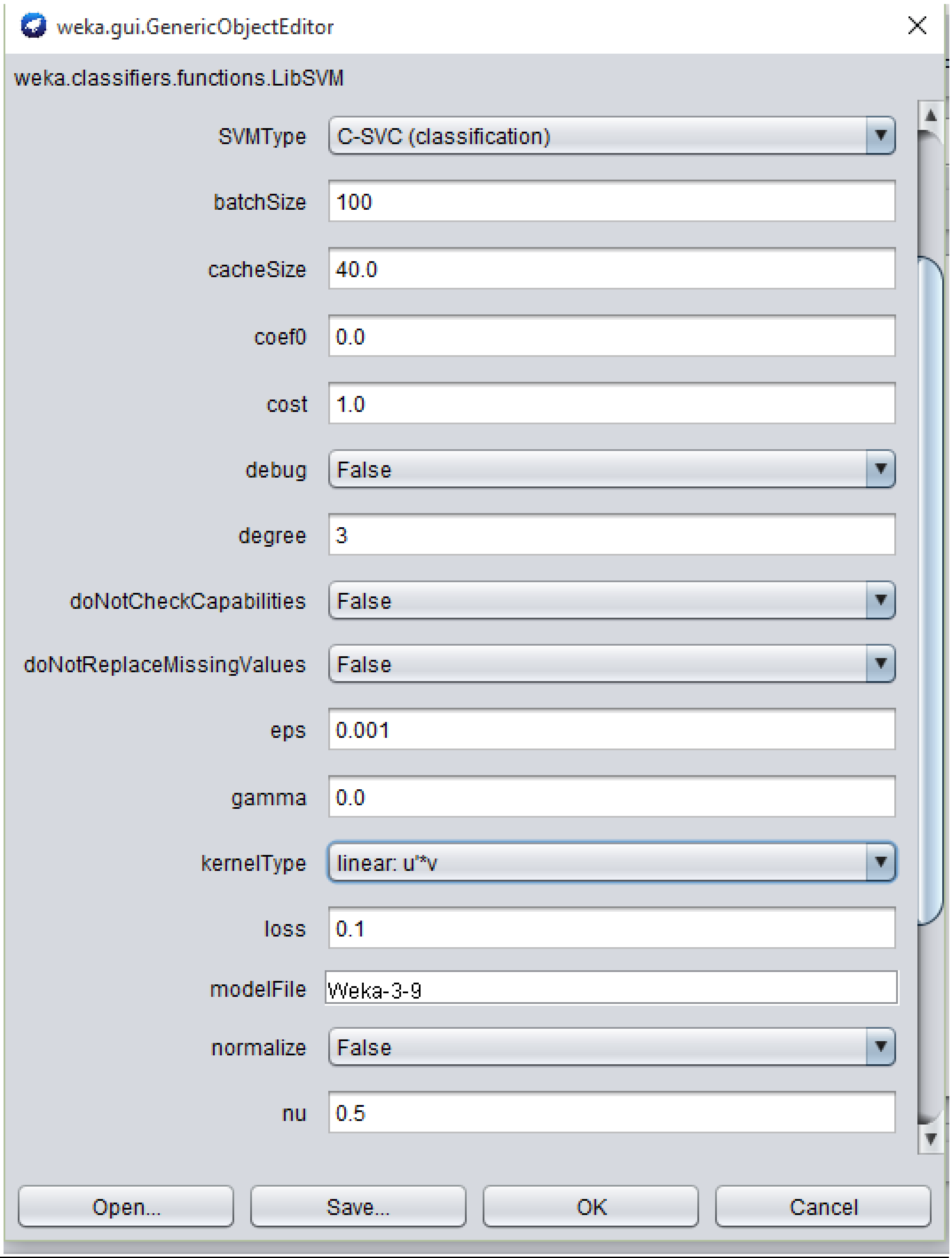
The performance metrics for the two classes assuming one positive and negative have given by the following formulae-

- **TP rate=** $\frac{TP}{TP+FN} = \frac{TP}{P}$ **FP rate =** $\frac{FP}{TN+FP}=\frac{FP}{N}$
- **PPV=** $\frac{TP}{TP+FP}$ **NPV=** $\frac{TN}{TN+FN}$
- **Specificity=** $\frac{TN}{TN+FP}$ **= 1-FP rate** **Sensitivity=** $\frac{TP}{TP+FN}$ **=TP rate**
- **Accuracy=** $\frac{TP+TN}{n}$**,** where n is the total number of samples
- **Geometric Mean =** $\sqrt{SE * SP}$
- **n=TP+FN+FP+TN**
- **Error rate=** $\frac{FP+FN}{n}$ **=1-Accuracy**

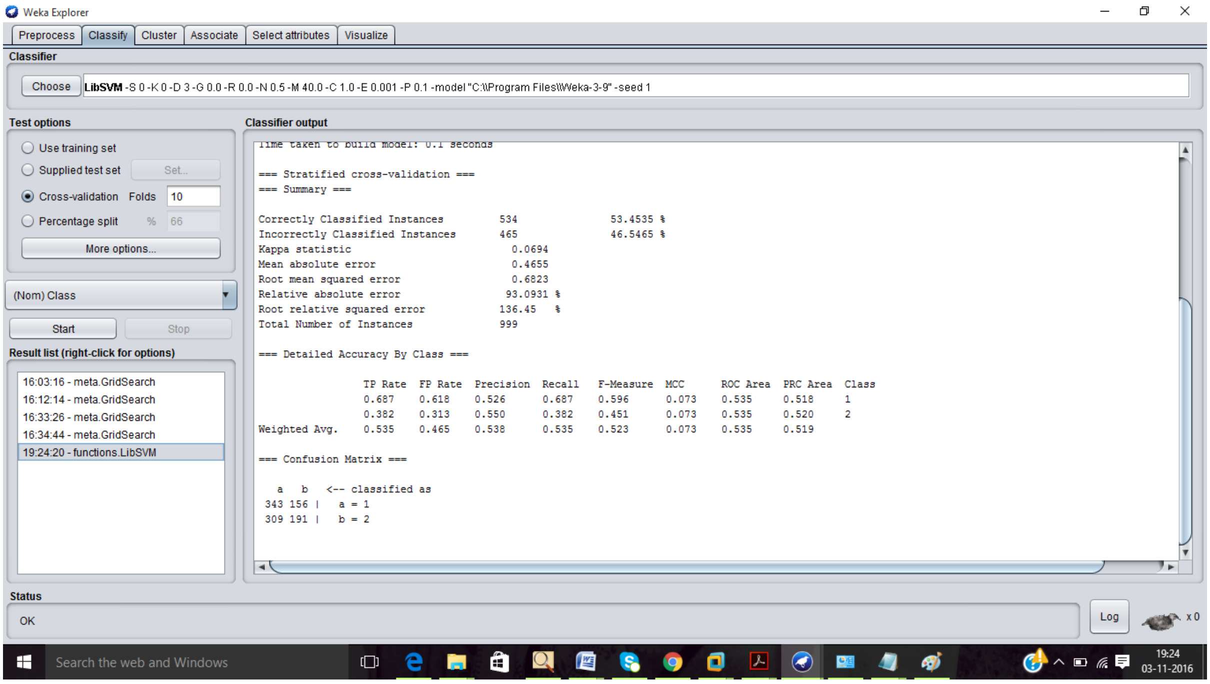
i) CLUSTER IN CLUSTER DATASET:-

<u>SVM- L(Linear kernel)</u>

<u>Screenshot: Choosing the Kernel Type as Linear and using defaultparameters after selecting LibSVM Classifier</u>



Screenshot: Efficiency measures for Linear Classifier



TP= 343 FN=156 FP= 309 TN=191

The total no of samples in this dataset is n=343+156+309+191=999

TP rate = $\frac{343}{343+156}$=0.687 FP rate= $\frac{309}{191+309}$=0.618

PPV = $\frac{343}{343+309}$=0.526 NPV=$\frac{191}{191+156}$=0.55

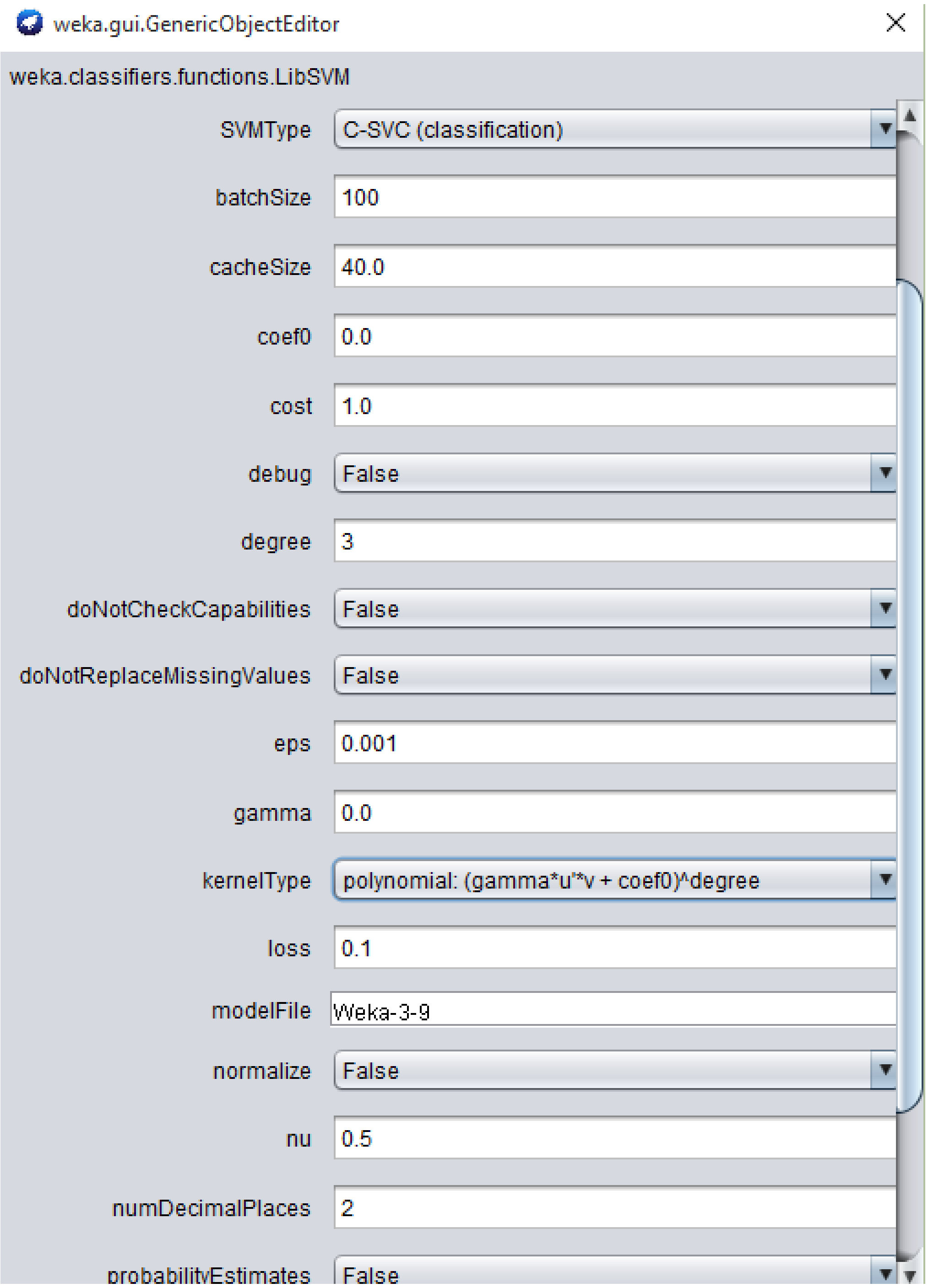
Specificity= 1- 0.618=0.382Sensitivity=0.687

Geometric Mean= $\sqrt{0.382 * 0.687}$=0.5122

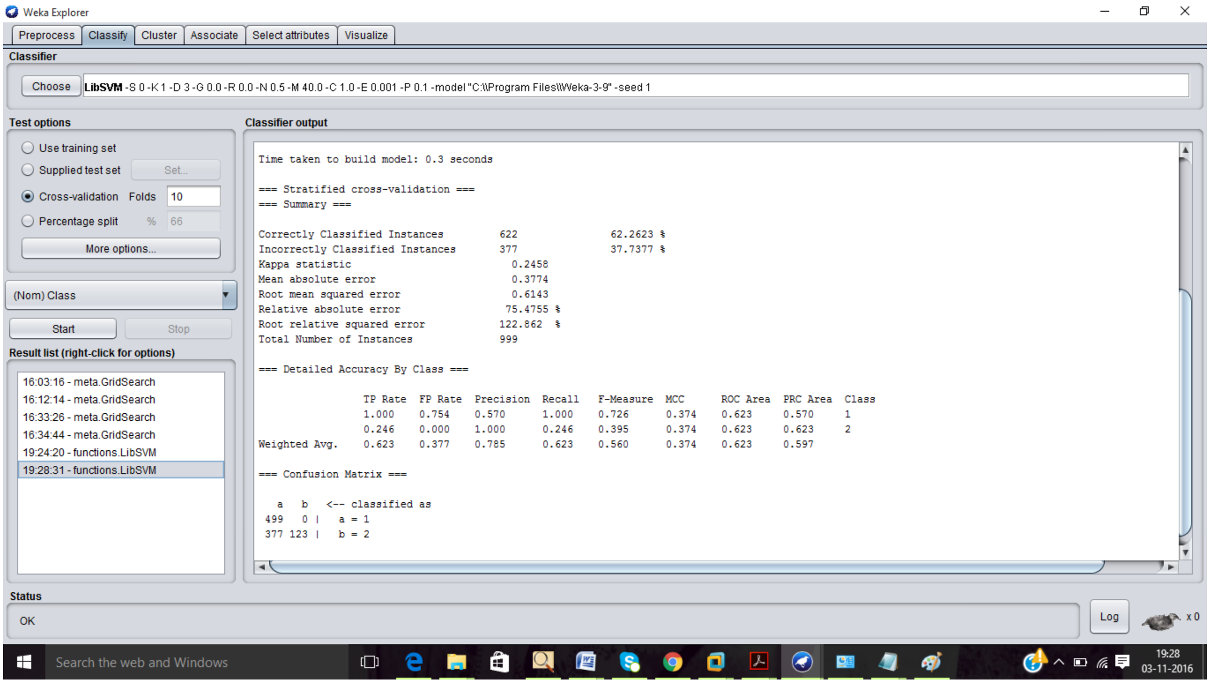
Accuracy=$\frac{343+191}{999}$=0.534=53.4% Error rate= 1-0.534=0.466=46.6%

<u>SVM –P (Polynomial kernel)</u>

<u>Screenshot: Choosing the Kernel Type as Polynomial and using default parameters after selecting LibSVM Classifier</u>



Screenshot: Efficiency measures for Polynomial Classifier with degree 3



TP=499 FN=0 FP= 377 TN=123

The total no of samples in this dataset is n=499+0+377+123=999

TP rate = $\frac{499}{499+0}$=1 FP rate= $\frac{377}{123+377}$=0.754

PPV = $\frac{499}{499+377}$= 0.569 NPV=$\frac{123}{123+0}$=1

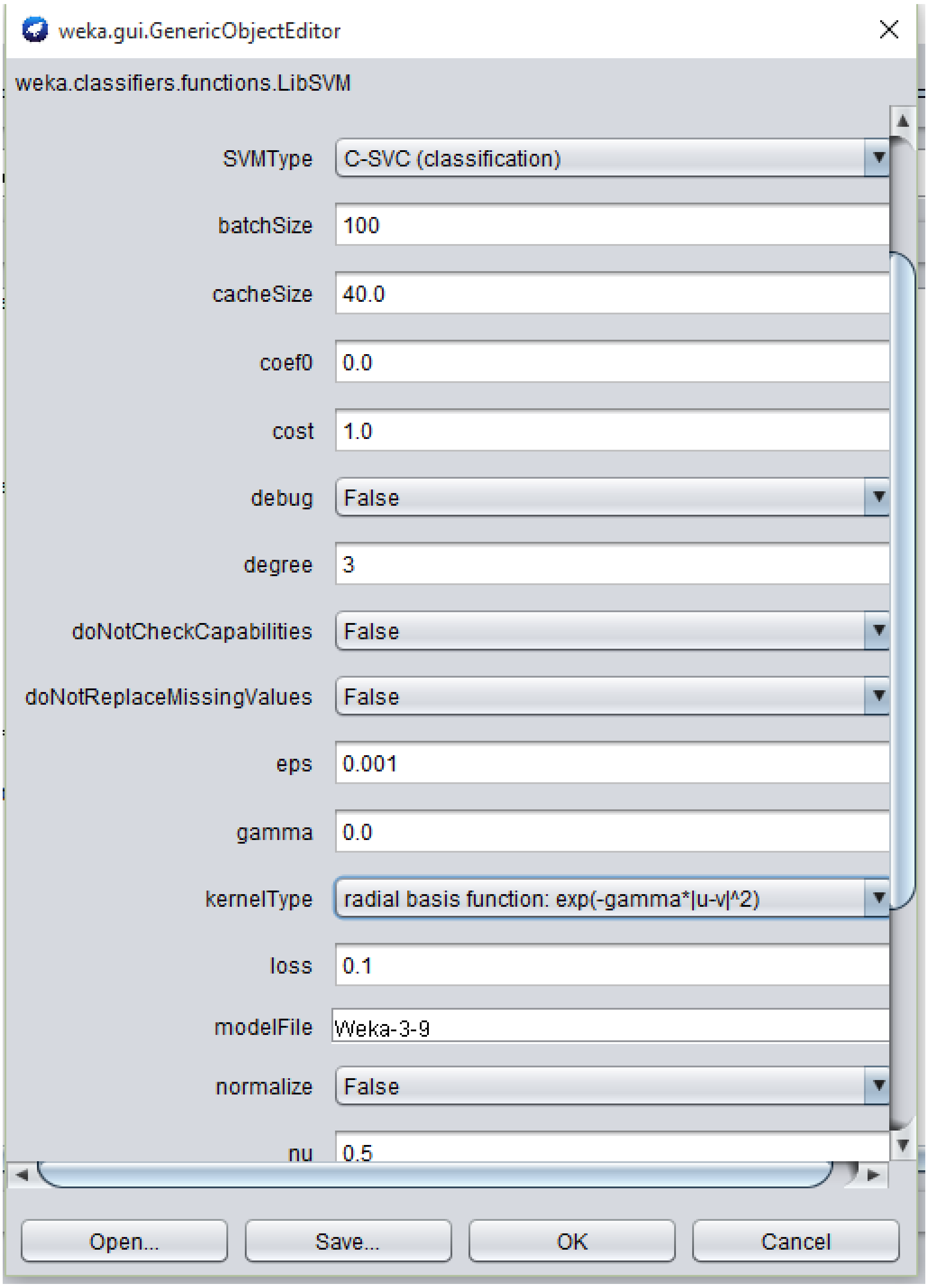
Specificity= 1- 0.754=0.246Sensitivity=1

Geometric Mean= $\sqrt{0.246 * 1}$=0.495

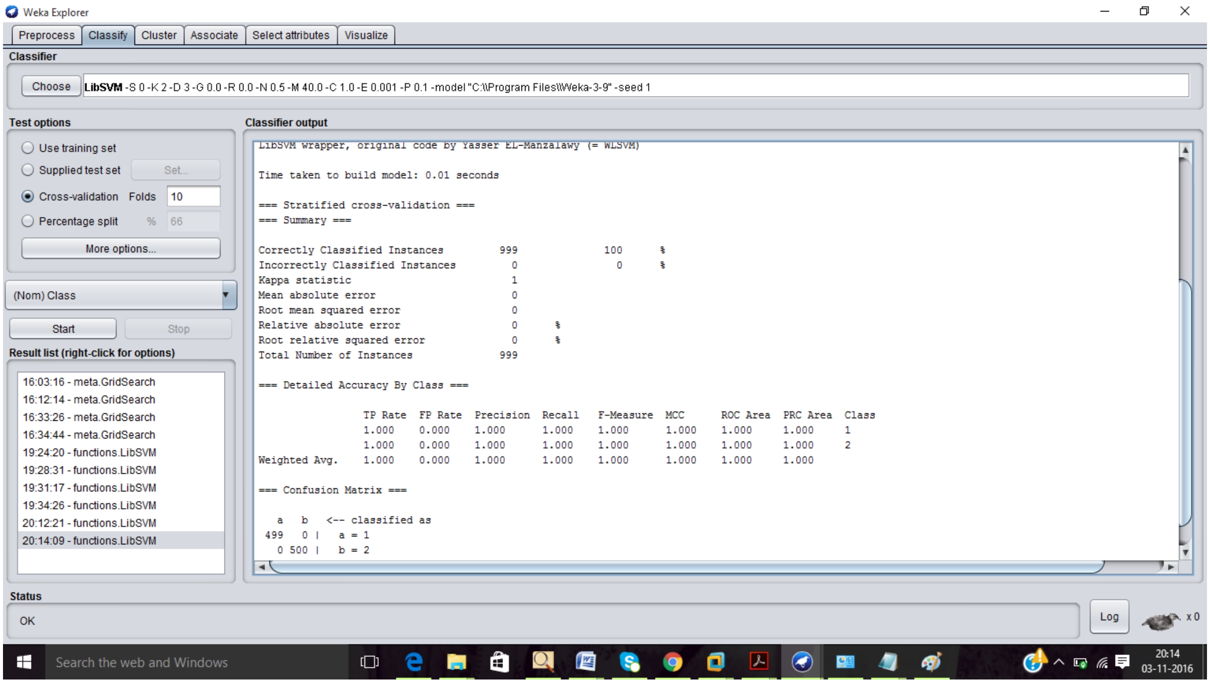
Accuracy= $\frac{499+123}{999}$=0.622=62.2% Error rate= 1-0.622=0.378=37.8%

<u>LibSVM-RBF</u>

<u>Screenshot: Choosing the Kernel Type as Radial Basis Function (RBF) and using default parameters after selecting LibSVM Classifier</u>



Screenshot: Efficiency measures for Radial Basis Classifier (RBF) Classifier



TP=499 FN=0 FP=0 TN=500

The total no of samples in this dataset is n=499+0+0+500=999

TP rate = $\frac{499}{499+0}$=1 FP rate= $\frac{0}{500+0}$=0.00

PPV = $\frac{499}{499+0}$=1 NPV=$\frac{500}{0+500}$=1

Specificity= 1- 0=1 Sensitivity=1

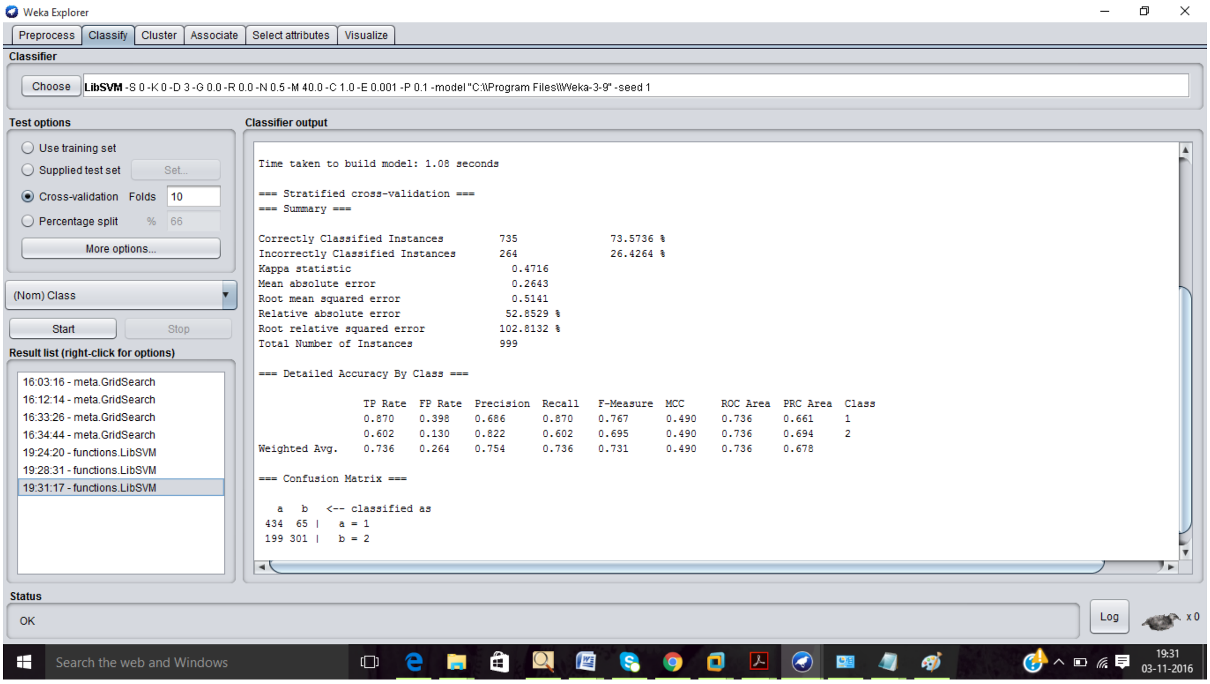
Geometric Mean= $\sqrt{1 * 1}$= 1

Accuracy= $\frac{499+500}{999}$=1=100% Error rate= 1-1=0=0%

ii) HALF KERNEL DATASET

<u>SVM- L(Linear kernel)</u>

<u>Screenshot: Efficiency measures for Linear Classifier</u>



TP=434 FN=65 FP=199 TN=301

The total no of samples in this dataset is n=434+65+199+301=999

TP rate = $\frac{434}{434+65}$=0.869 FP rate= $\frac{199}{301+199}$=0.398

PPV = $\frac{434}{434+199}$=0.685 NPV=$\frac{301}{301+65}$=0.822

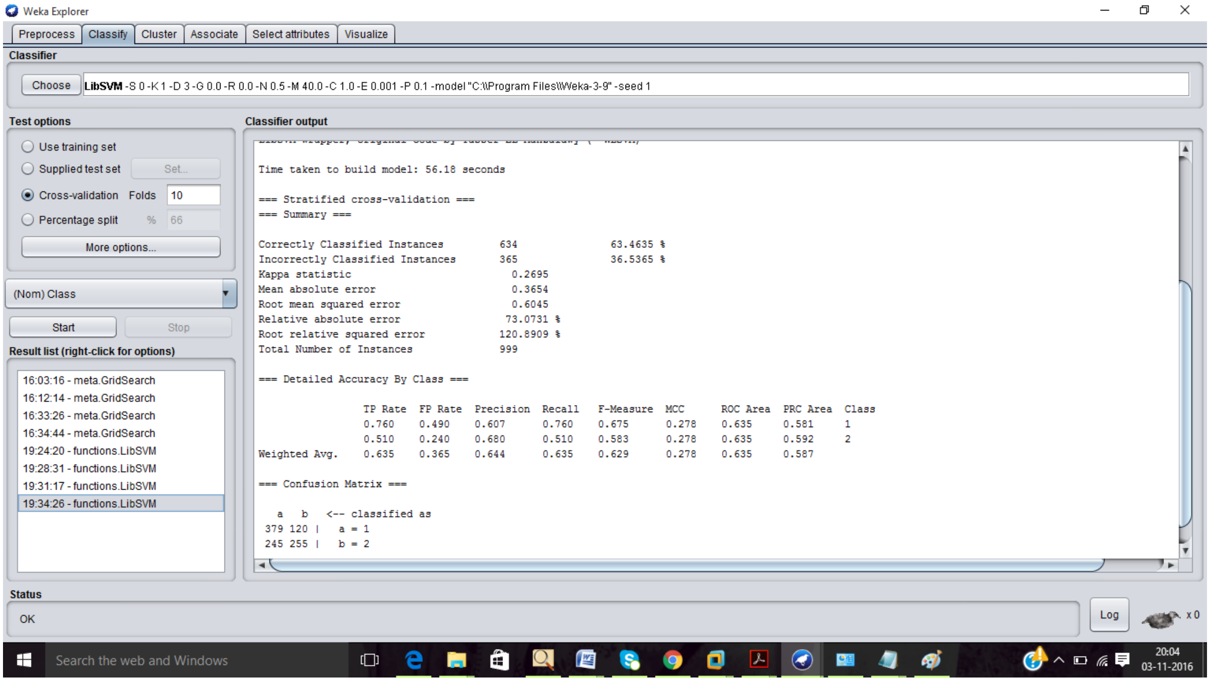
Specificity= 1- 0.398=0.602Sensitivity=0.869

Geometric Mean= $\sqrt{0.602 * 0.869}$=0.723

Accuracy= $\frac{434+301}{999}$=0.7357=73.57% Error rate= 1-0.7357=0.2643=26.43%

<u>SVM –P (Polynomial kernel)</u>

<u>Screenshot: Efficiency measures for Polynomial Classifier with degree 3</u>



TP= 379 FN=120 FP= 245 TN=255

The total no of samples in this dataset is n=379+120+245+255=999

TP rate = $\frac{379}{379+120}$=0.7595 FP rate= $\frac{245}{245+255}$=0.49

PPV = $\frac{379}{379+245}$=0.607 NPV=$\frac{255}{255+120}$=0.68

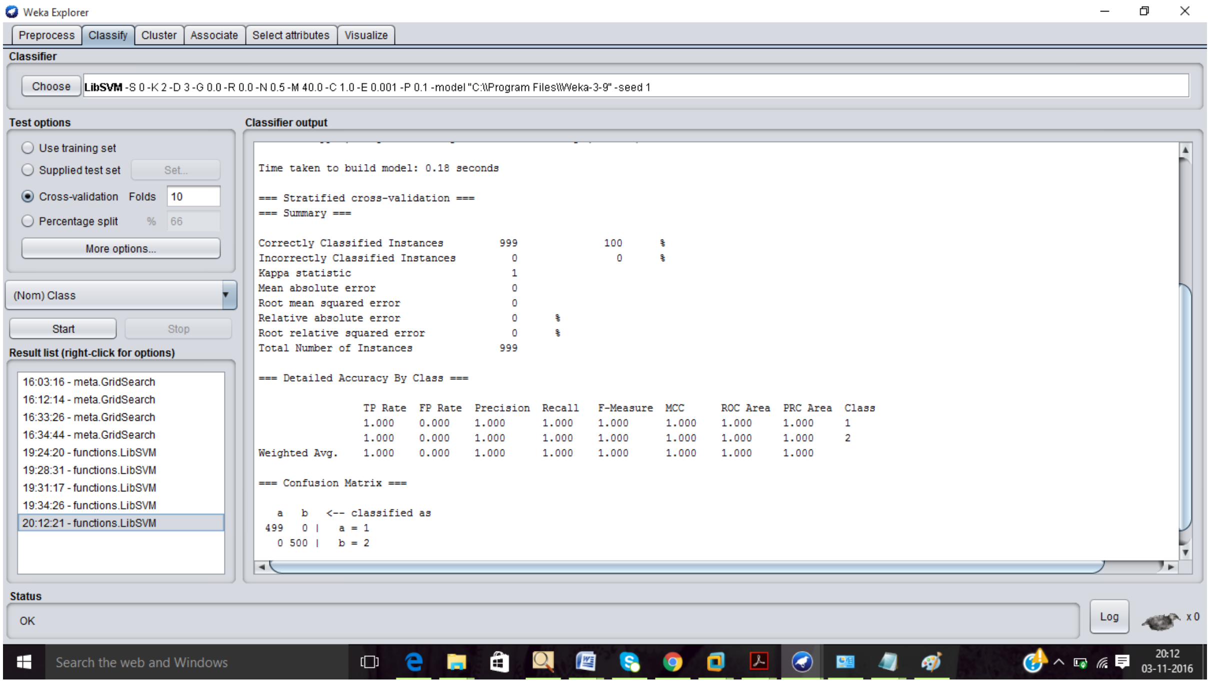
Specificity= 1- 0.49=0.51Sensitivity=0.7595

Geometric Mean= $\sqrt{0.51 * 0.7595}$=0.6223

Accuracy=$\frac{379+255}{999}$=0.634=63.4% Error rate= 1-0.634=0.366=36.6%

<u>LibSVM-RBF</u>

<u>Screenshot: Efficiency measures for Radial Basis Classifier (RBF) Classifier</u>



TP=499 FN=0 FP=0 TN=500

The total no of samples in this dataset is n=499+0+0+500=999

TP rate = $\frac{499}{499+0}$=1 FP rate= $\frac{0}{500+0}$=0.00

PPV = $\frac{499}{499+0}$=1 NPV=$\frac{500}{0+500}$=1

Specificity= 1- 0=1 Sensitivity=1

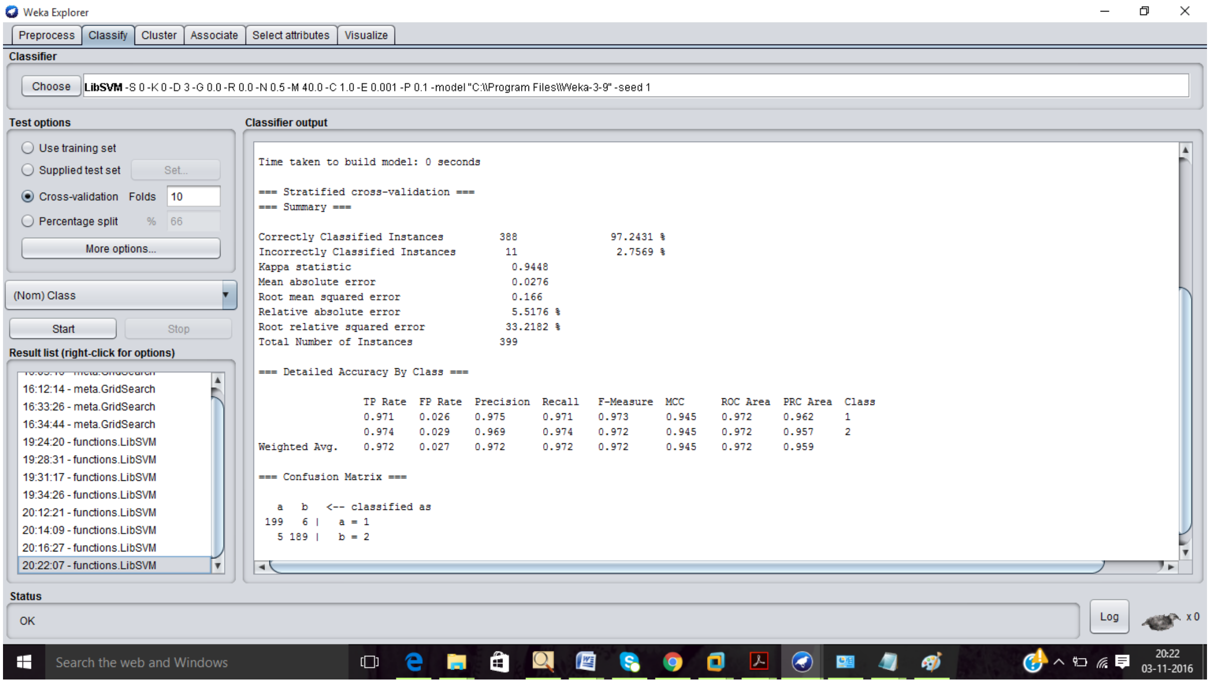
Geometric Mean= $\sqrt{1 * 1}$= 1

Accuracy= $\frac{499+500}{999}$=1=100% Error rate= 1-1=0=0%

iii) TWO GAUSSIANS DATASET

<u>SVM- L(Linear kernel)</u>

<u>Screenshot: Efficiency measures for Linear Classifier</u>



TP=199 FN=6 FP=5 TN=189

The total no of samples in this dataset is n=199+6+5+189=399

TP rate = $\frac{199}{199+6}$=0.97 FP rate= $\frac{5}{189+5}$=0.025

PPV = $\frac{199}{199+5}$=0.975 NPV=$\frac{189}{189+6}$=0.9692

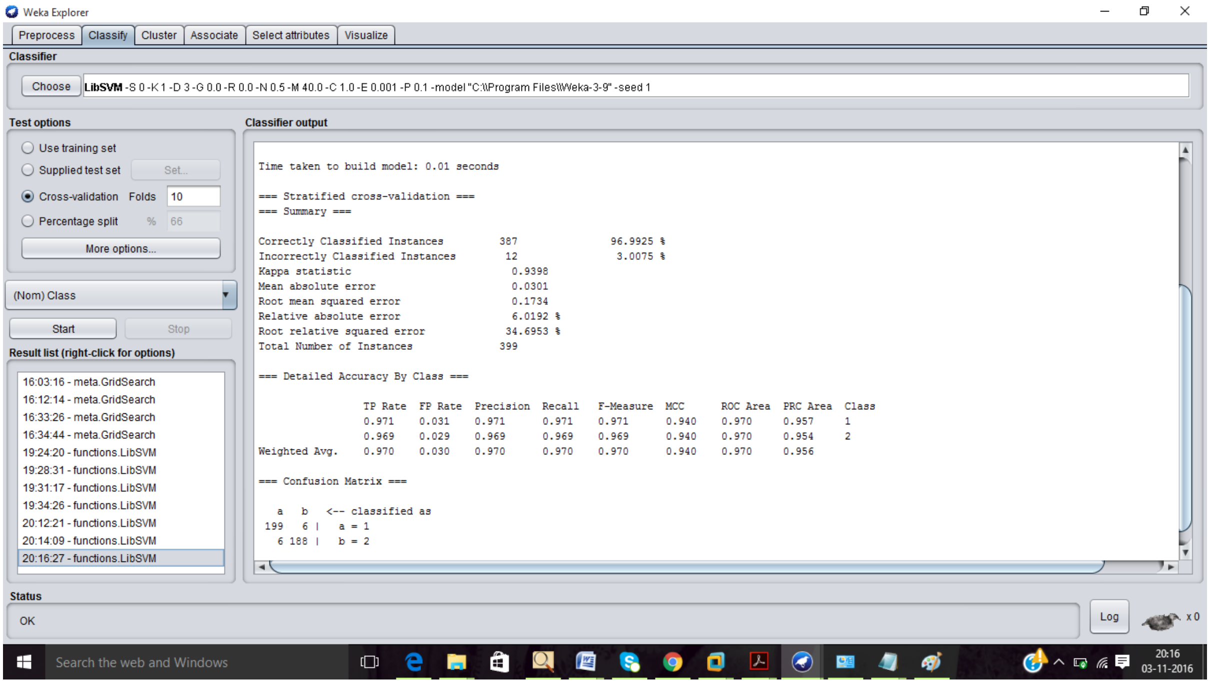
Specificity= 1-0.025=0.975Sensitivity=0.97

Geometric Mean= $\sqrt{0.975 * 0.97}$=0.972

Accuracy= $\frac{199+189}{399}$=0.972=97.2% Error rate= 1-0.972=0.028=2.8%

<u>SVM –P (Polynomial kernel)</u>

<u>Screenshot: Efficiency measures for Polynomial Classifier with degree 3</u>



TP=199 FN=6 FP=6 TN=188

The total no of samples in this dataset is n=199+6+6+188=399

TP rate = $\frac{199}{199+6}$=0.97 FP rate= $\frac{6}{6+188}$=0.03

PPV = $\frac{199}{199+6}$=0.97 NPV=$\frac{188}{188+6}$=0.9690

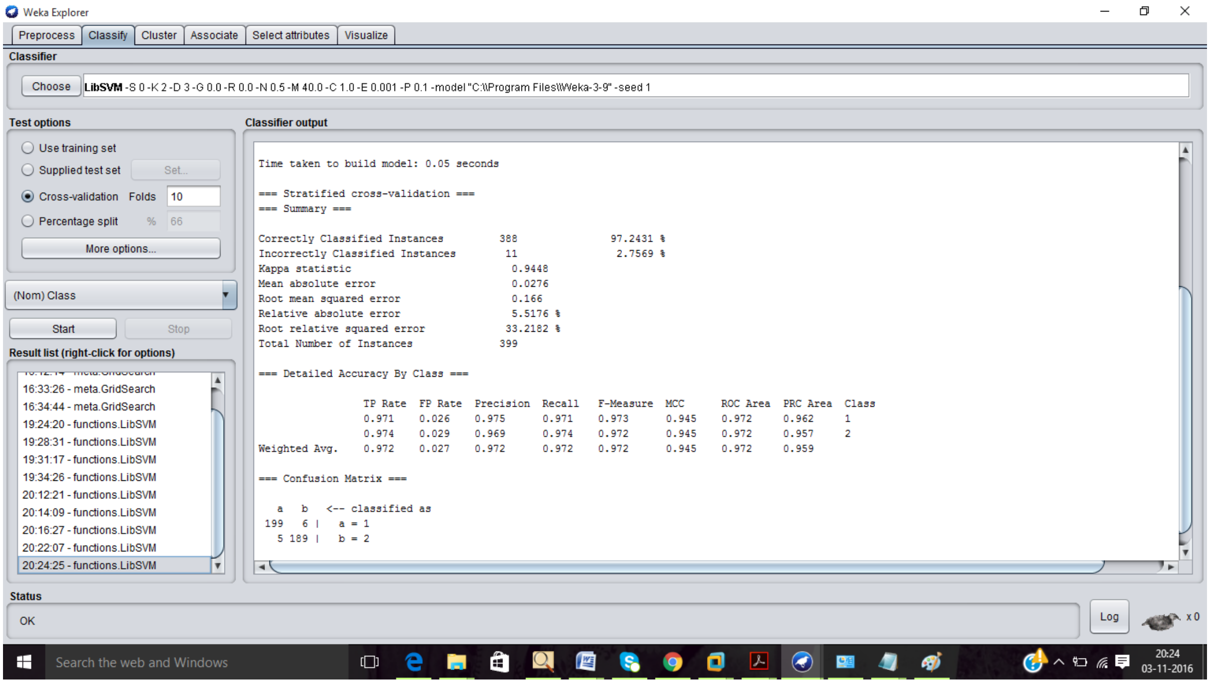
Specificity= 1-0.03=0.97Sensitivity=0.97

Geometric Mean= $\sqrt{0.97 * 0.97}$=0.97

Accuracy= $\frac{199+188}{399}$=0.9699=96.99% Error rate= 1-0.9699=0.0301=3.01%

LibSVM-RBF

Screenshot: Efficiency measures for Radial Basis Classifier (RBF) Classifier



TP=199 FN=6 FP=5 TN=189

The total no of samples in this dataset is n=199+6+5+189=399

TP rate = $\frac{199}{199+6}$=0.97 FP rate= $\frac{5}{189+5}$=0.025

PPV = $\frac{199}{199+5}$=0.975 NPV=$\frac{189}{189+6}$=0.9692

Specificity= 1-0.025=0.975Sensitivity=0.97

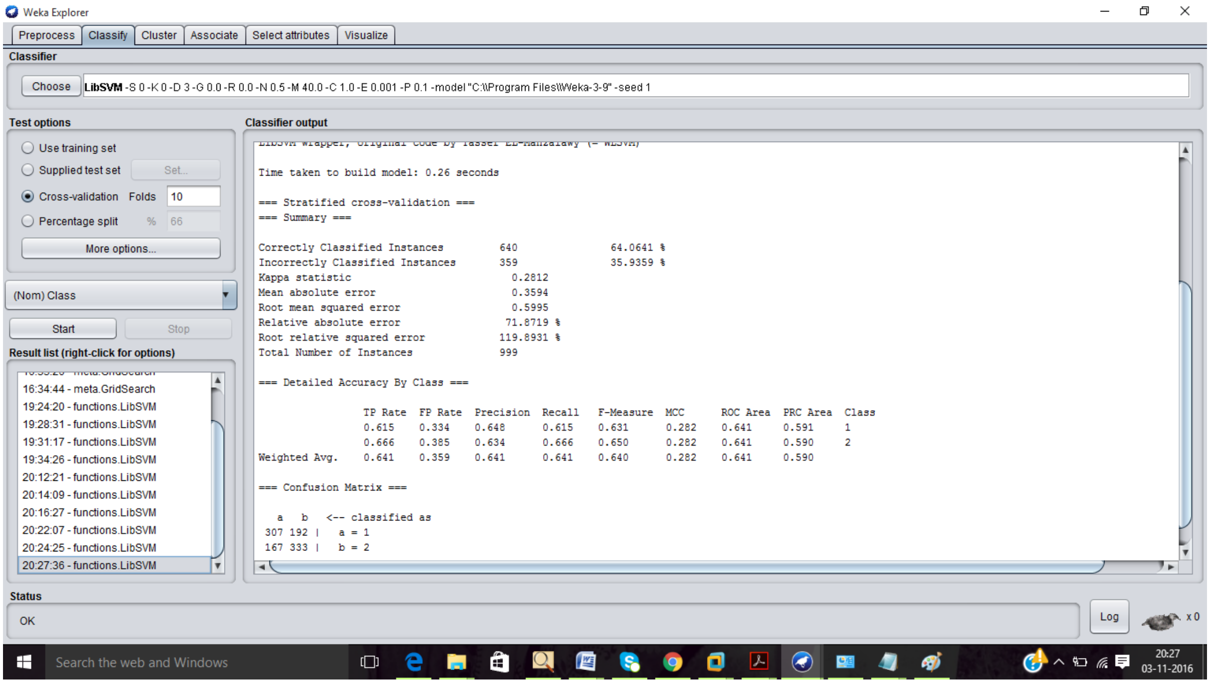
Geometric Mean= $\sqrt{0.975 * 0.97}$=0.972

Accuracy=$\frac{199+189}{399}$=0.972=97.2% Error rate= 1-0.972=0.028=2.8%

iv) TWO SPIRALS DATASET:

<u>SVM- L(Linear kernel)</u>

<u>Screenshot: Efficiency measures for Linear Classifier</u>



TP= 307 FN=192 FP=167 TN=333

The total no of samples in this dataset is n=307+192+167+333=999

TP rate = $\frac{307}{307+192}$=0.615 FP rate= $\frac{167}{333+167}$=0.334

PPV = $\frac{307}{307+167}$=0.6476 NPV=$\frac{333}{333+167}$=0.666

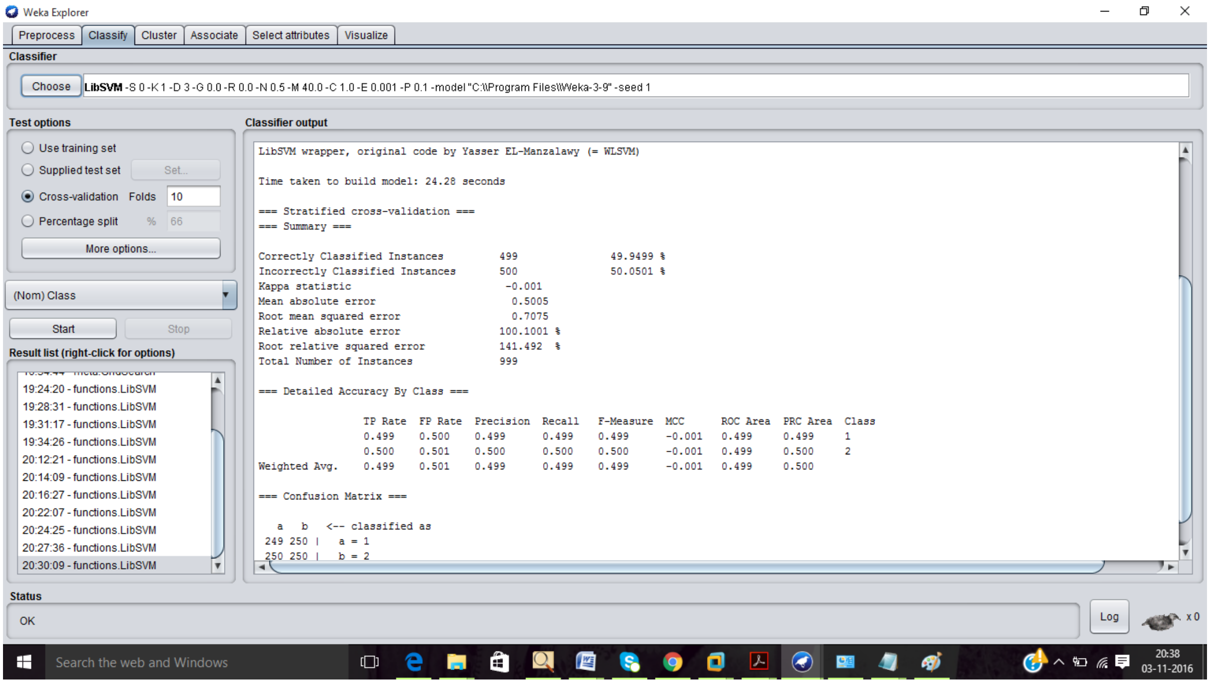
Specificity= 1- 0.334=0.666 Sensitivity=0.615

Geometric Mean= $\sqrt{0.666 * 0.615}$=0.6399

Accuracy= $\frac{307+333}{999}$=0.6406=64.06% Error rate= 1-0.6406=0.3594=35.94%

<u>SVM –P (Polynomial kernel)</u>

<u>Screenshot: Efficiency measures for Polynomial Classifier with degree 3</u>



TP=249 FN=250 FP=250 TN=250

The total no of samples in this dataset is n=249+250+250+250=999

TP rate = $\frac{249}{249+250}$=0.498 FP rate= $\frac{250}{250+250}$=0.5

PPV = $\frac{249}{249+250}$=0.498 NPV=$\frac{250}{250+250}$=0.5

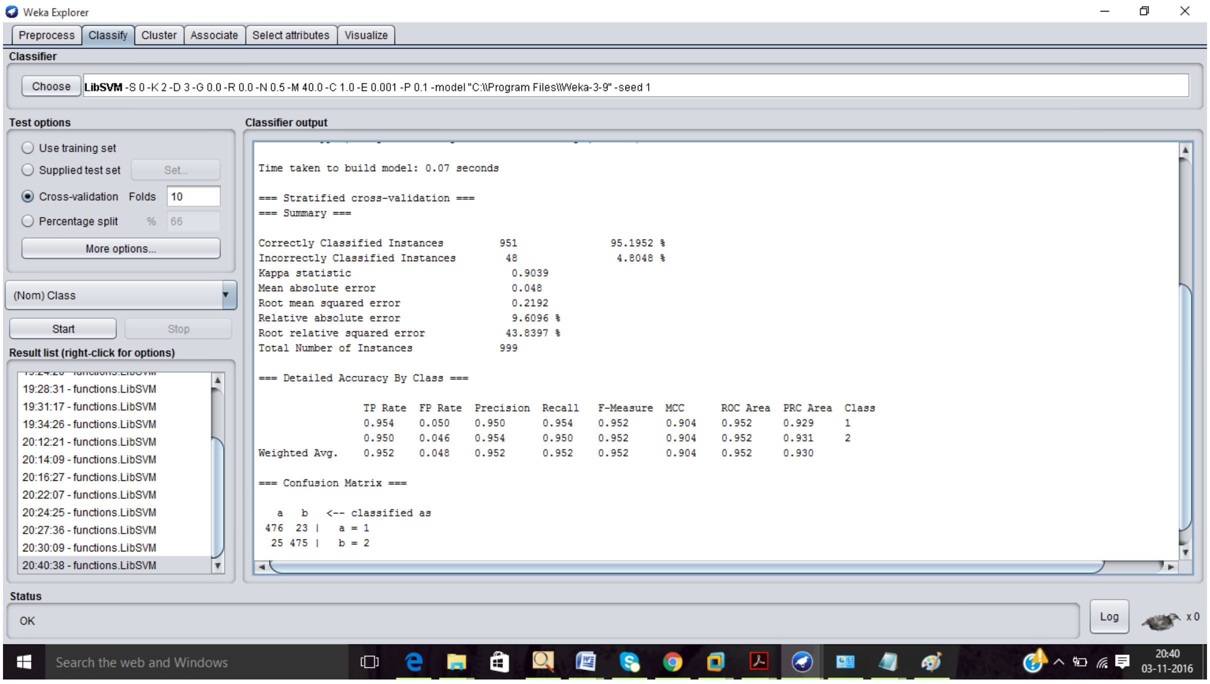
Specificity= 1- 0.5=0.5 Sensitivity=0.498

Geometric Mean= $\sqrt{0.5 * 0.498}$=0.498

Accuracy=$\frac{249+250}{999}$=0.4994=49.94% Error rate= 1-0.4994=0.5006=50.06%

<u>LibSVM-RBF</u>

<u>Screenshot: Efficiency measures for Radial Basis Classifier (RBF) Classifier</u>



TP= 476 FN=23 FP= 25 TN=475

The total no of samples in this dataset is n=476+23+25+475=999

TP rate = $\frac{476}{476+23}$=0.9539 FP rate= $\frac{25}{25+475}$=0.05

PPV = $\frac{476}{476+25}$=0.95 NPV=$\frac{475}{475+23}$=0.9538

Specificity= 1- 0.05=0.95Sensitivity=0.9539

Geometric Mean= $\sqrt{0.95 * 0.9539}$=0.9519

Accuracy=$\frac{476+475}{999}$=0.9519=95.19% Error rate= 1-0.9519=0.0481=4.81%

4) **GRID SEARCH for each dataset**

Grid Search is used for optimizing the parameters of the SVM kernel. It tunes and finds the value of Cost (C), gamma (σ) and degree(d). The parameters vary from:

d -> degree of the polynomial ranges from 2 to 8

C -> Cost ranges from 1 to 10^6

σ –> Gamma ranges from 0.01 to 2^8

These parameters are data dependent as they entirely vary based on the dataset. The grid search is performed on 3D or higher space after the mapping.

The parameters are tuned by applying normalisation (scaling of the data to [0,1] or [-1,1]), grid search and cross validation. In Weka, set normalize to ‘True’ to normalize all the data.

The best accuracy is obtained after varying these parameters. Based on the kernel, vary the parameters as-

- SVM-Linear: Evaluate Cost to find optimized accuracy.

- SVM-Polynomial: Evaluate Gamma and Degree in X and Y axes to find optimized accuracy.
- SVM-RBF: Evaluate Cost and Gamma in X and Y axes to find optimized accuracy.

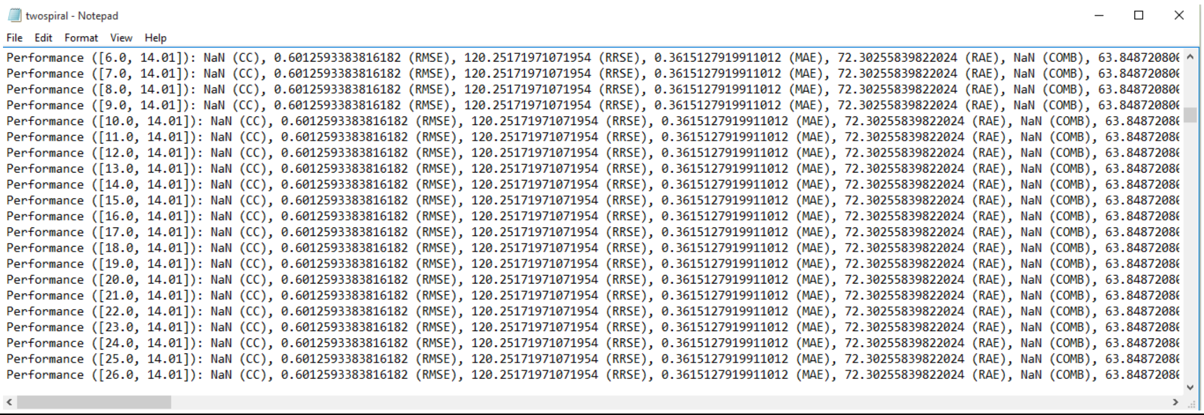
In Weka, choose Kernel Type as Linear, Polynomial or RBF to classify the samples.

i) CLUSTER IN CLUSTER DATASET

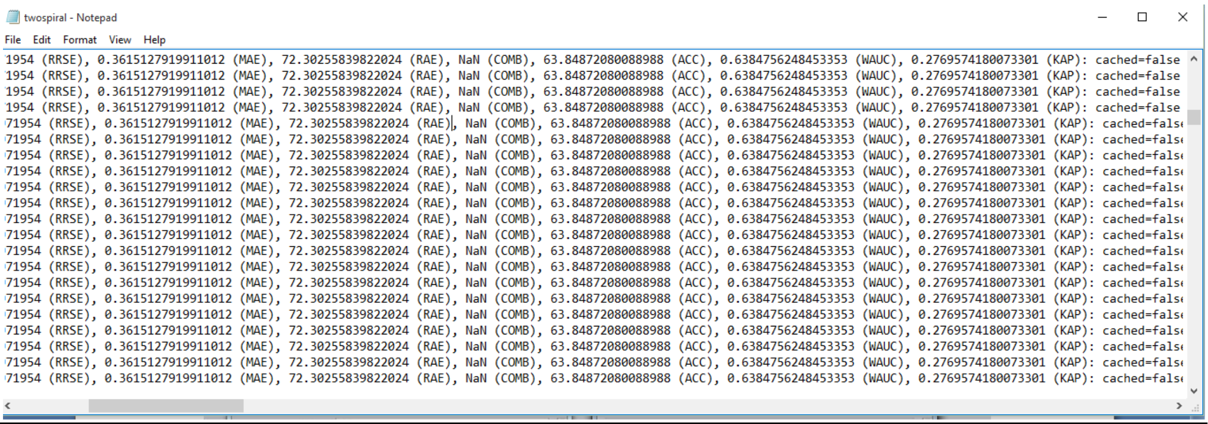
SVM- L(Linear kernel)

The accuracy is increased at certain parameter values while optimising it.

Screenshot: Log files captured while performing grid search



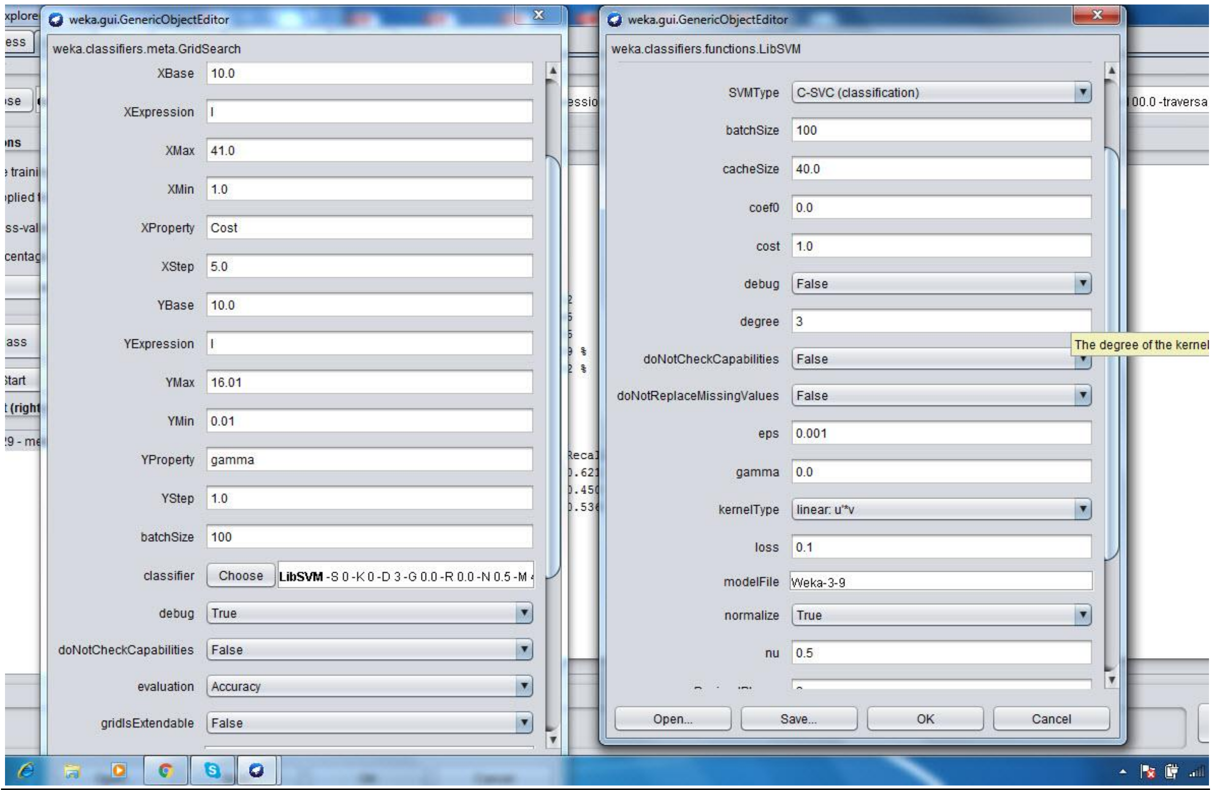
```
twospiral - Notepad
File Edit Format View Help
Performance ([6.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080
Performance ([7.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080
Performance ([8.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080
Performance ([9.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080
Performance ([10.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([11.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([12.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([13.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([14.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([15.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([16.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([17.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([18.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([19.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([20.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([21.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([22.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([23.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([24.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([25.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
Performance ([26.0, 14.01]): NaN (CC), 0.6012593383816182 (RMSE), 120.25171971071954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.8487208
```



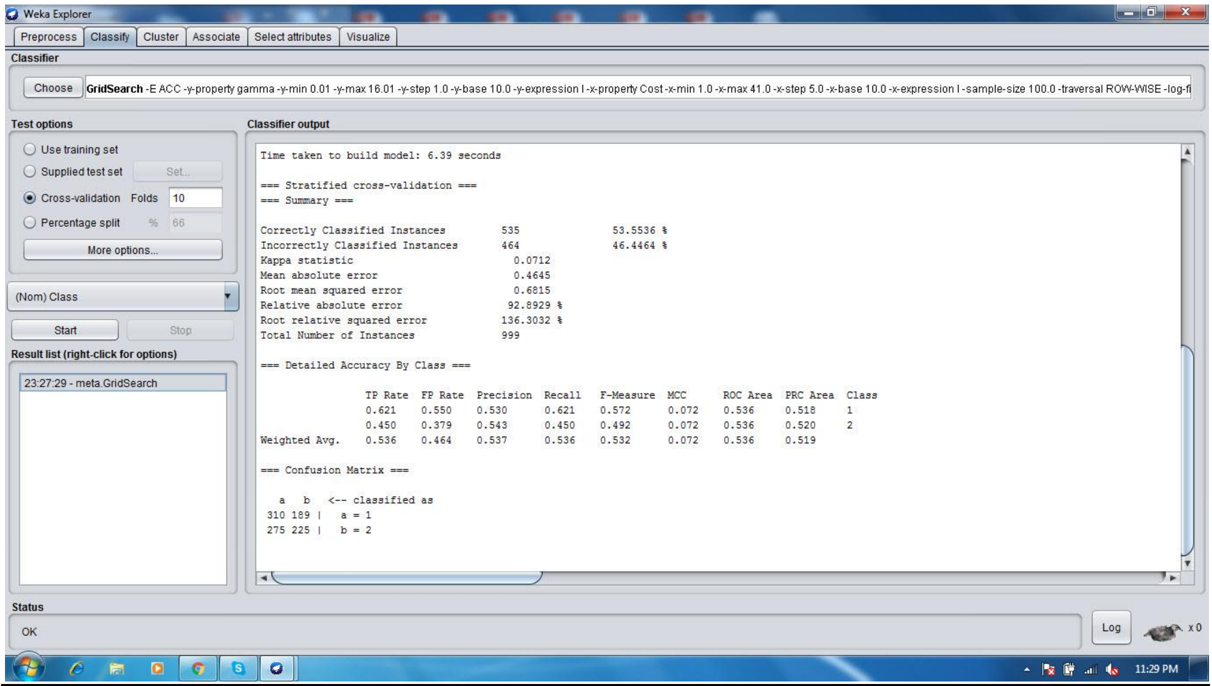
```
twospiral - Notepad
File Edit Format View Help
1954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=false
1954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=false
1954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=false
1954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=false
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
71954 (RRSE), 0.3615127919911012 (MAE), 72.30255839822024 (RAE), NaN (COMB), 63.84872080088988 (ACC), 0.6384756248453353 (WAUC), 0.2769574180073301 (KAP): cached=fals
```

While debugging the grid search, the steps being executed are logged in the log file. For different parameters of gamma, cost and degree, the accuracy varies. It runs for all the step sizes (iterations) from maximum to minimum values of X and Y parameters.

Screenshot: Tuning the parameters of SVM Linear Kernel using grid search

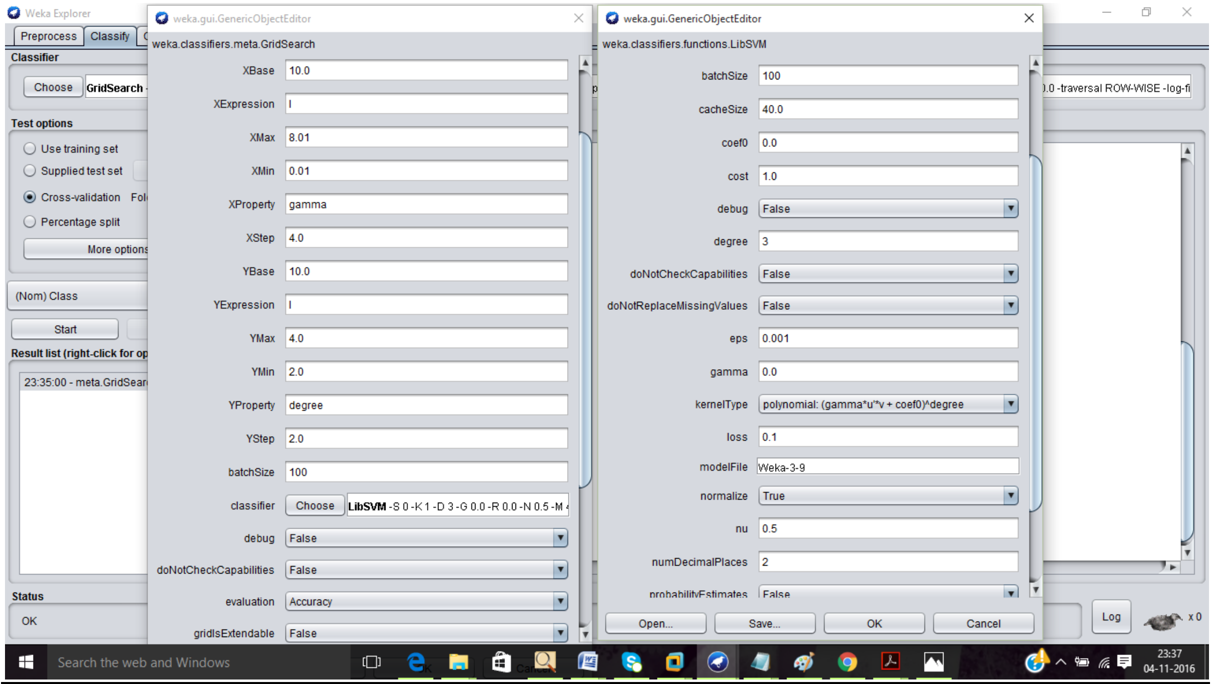


Screenshot: Results optimization using grid search for linear kernel



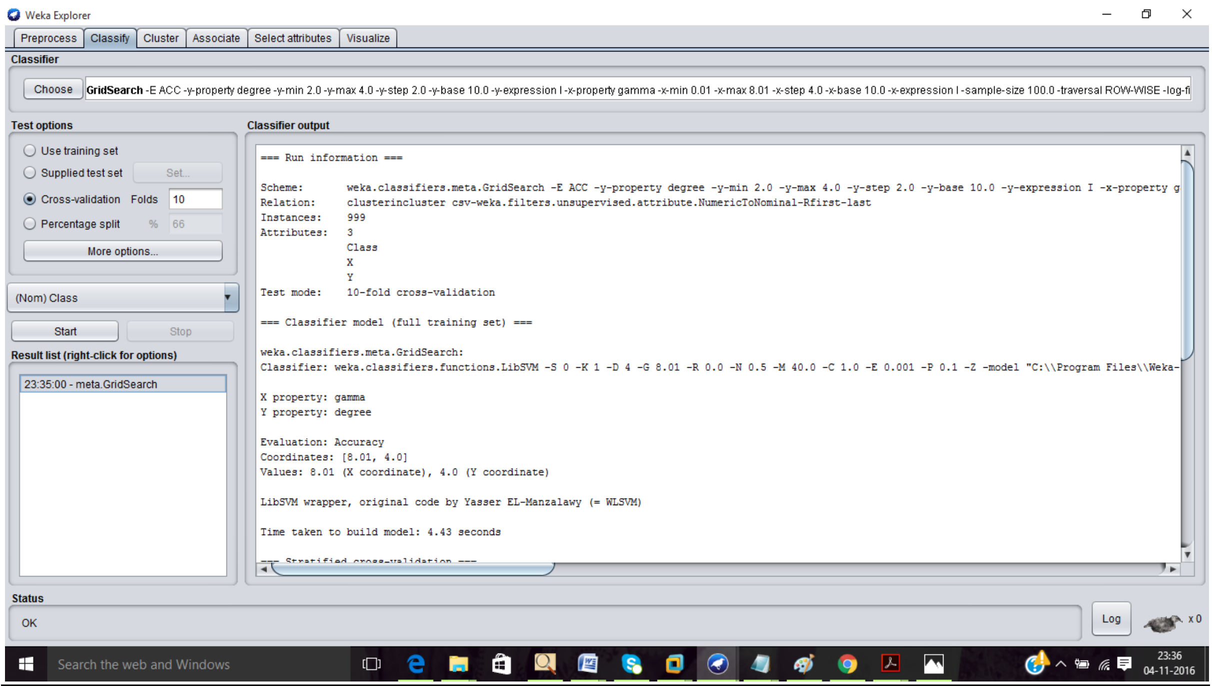
SVM –P (Polynomial kernel)

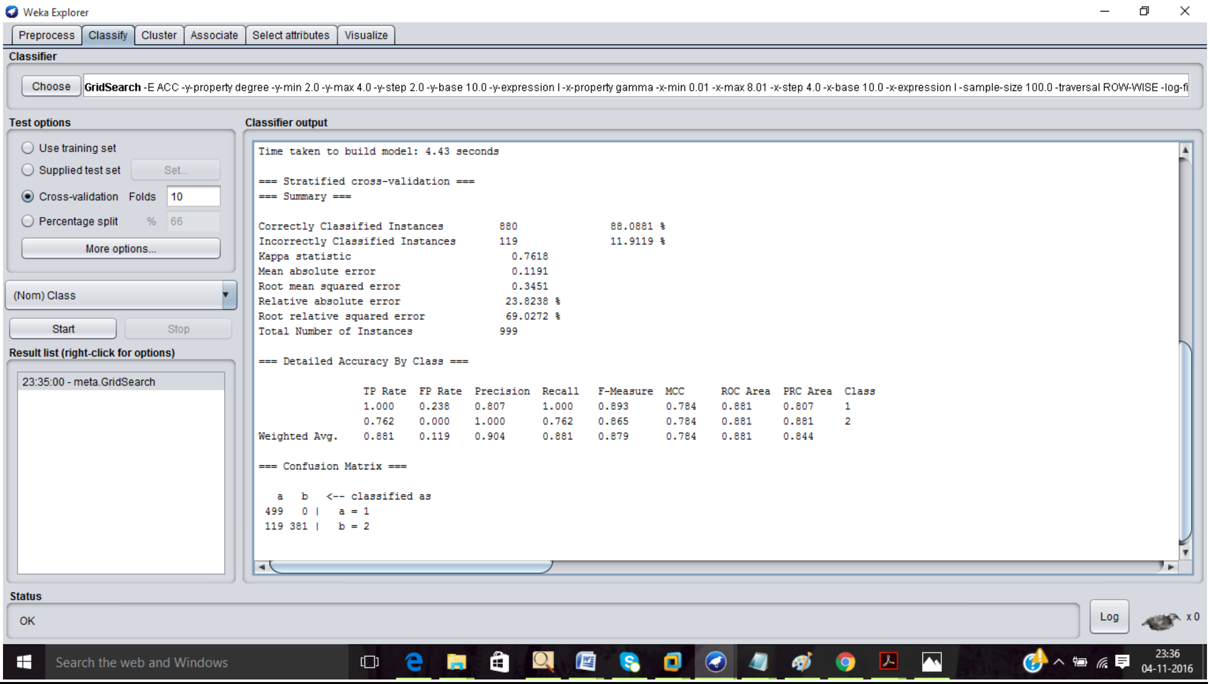
Screenshot: Tuning the parameters of SVM Polynomial Kernel using grid search



Screenshot: Results optimization using grid search for Polynomial kernel

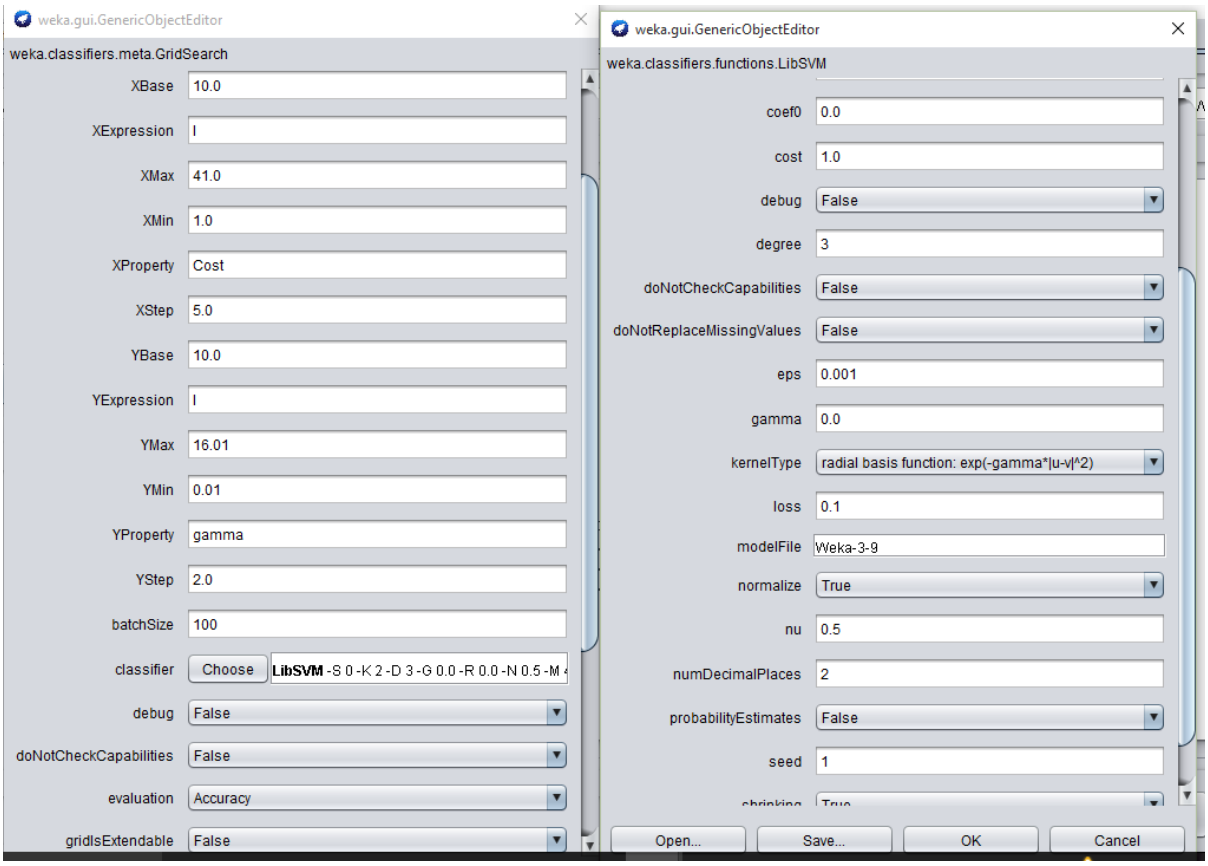
The accuracy obtained is 88.0881% by varying parameters (gamma=8.01,degree=4). By using grid search, the accuracy is optimized to 88.0881% than using default parameters in cross validation which produced accuracy of 62.2623%.



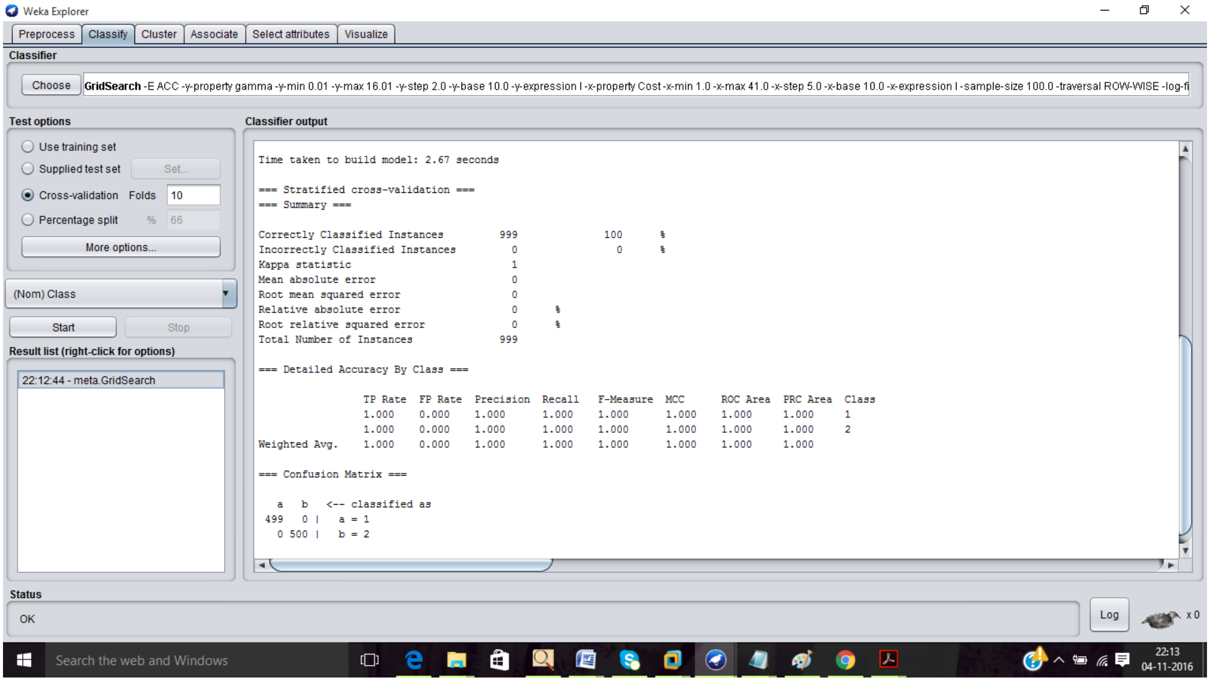


SVM- R (Radial Base Function kernel)

Screenshot: Tuning the parameters of SVM RBF Kernel using grid search



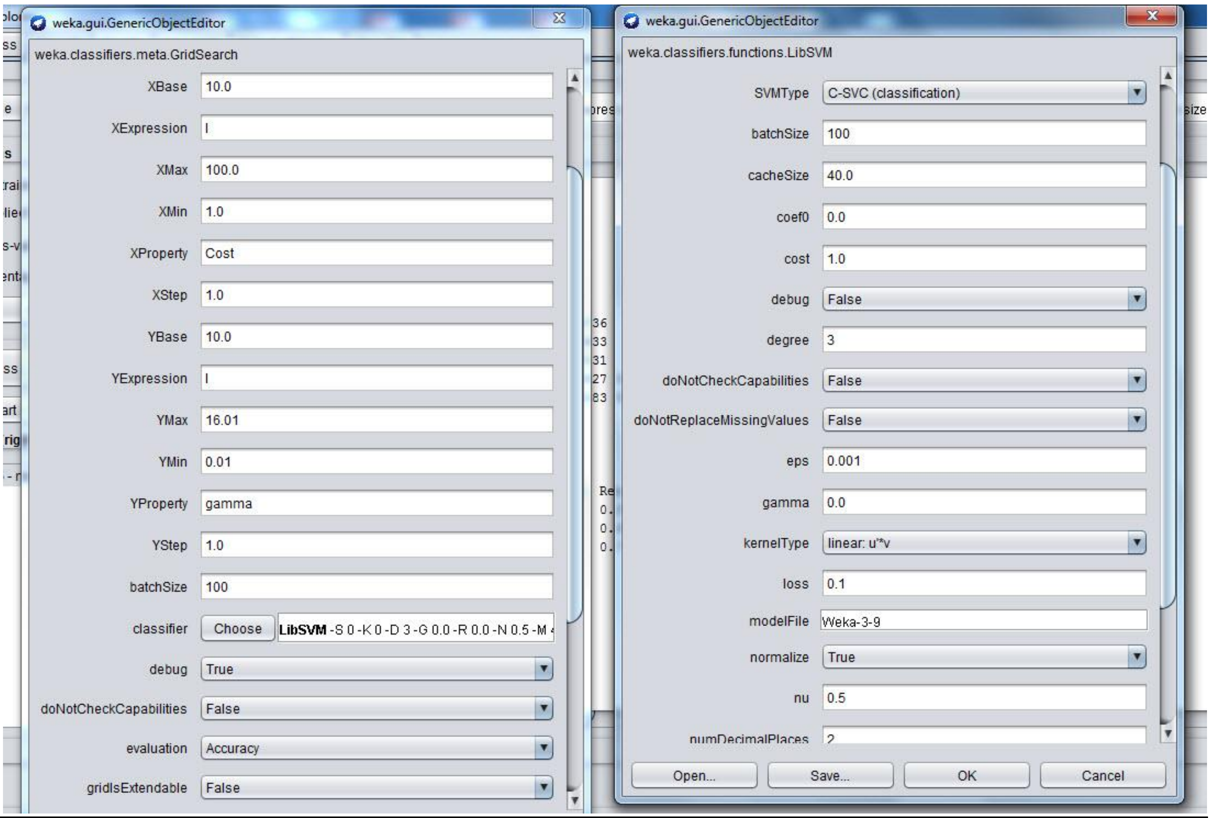
Screenshot: Results optimization using grid search for RBF kernel



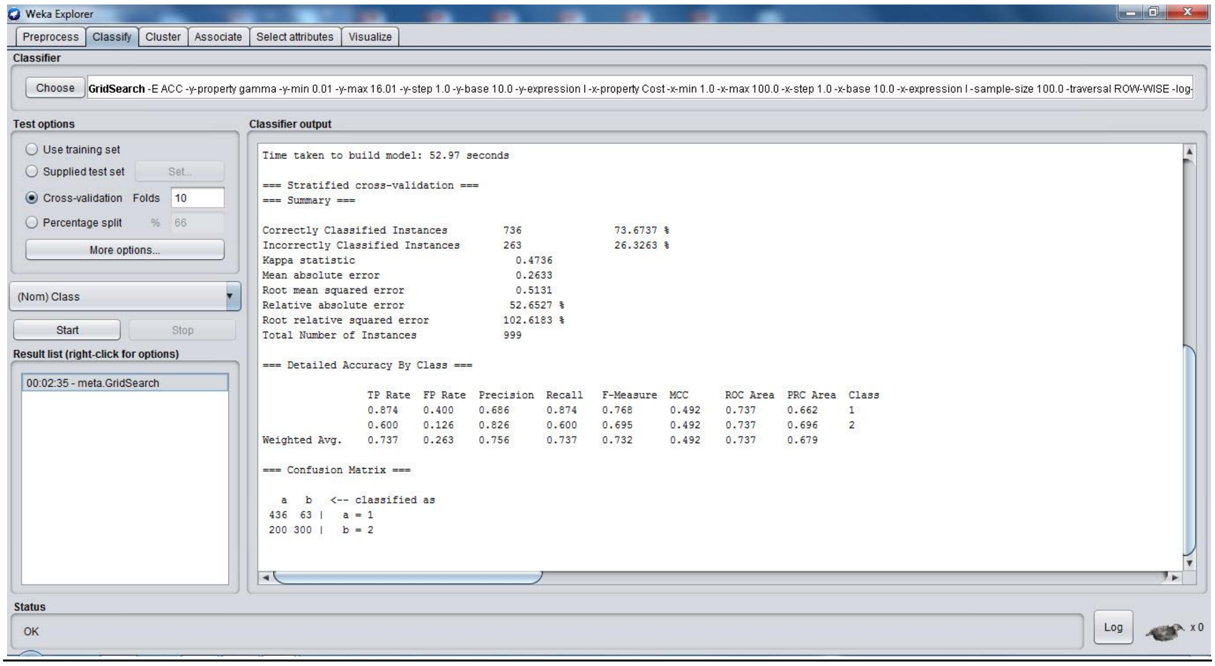
ii) HALF KERNAL DATASET

SVM- L (Linear kernel)

Screenshot: Tuning the parameters of SVM Linear Kernel using grid search

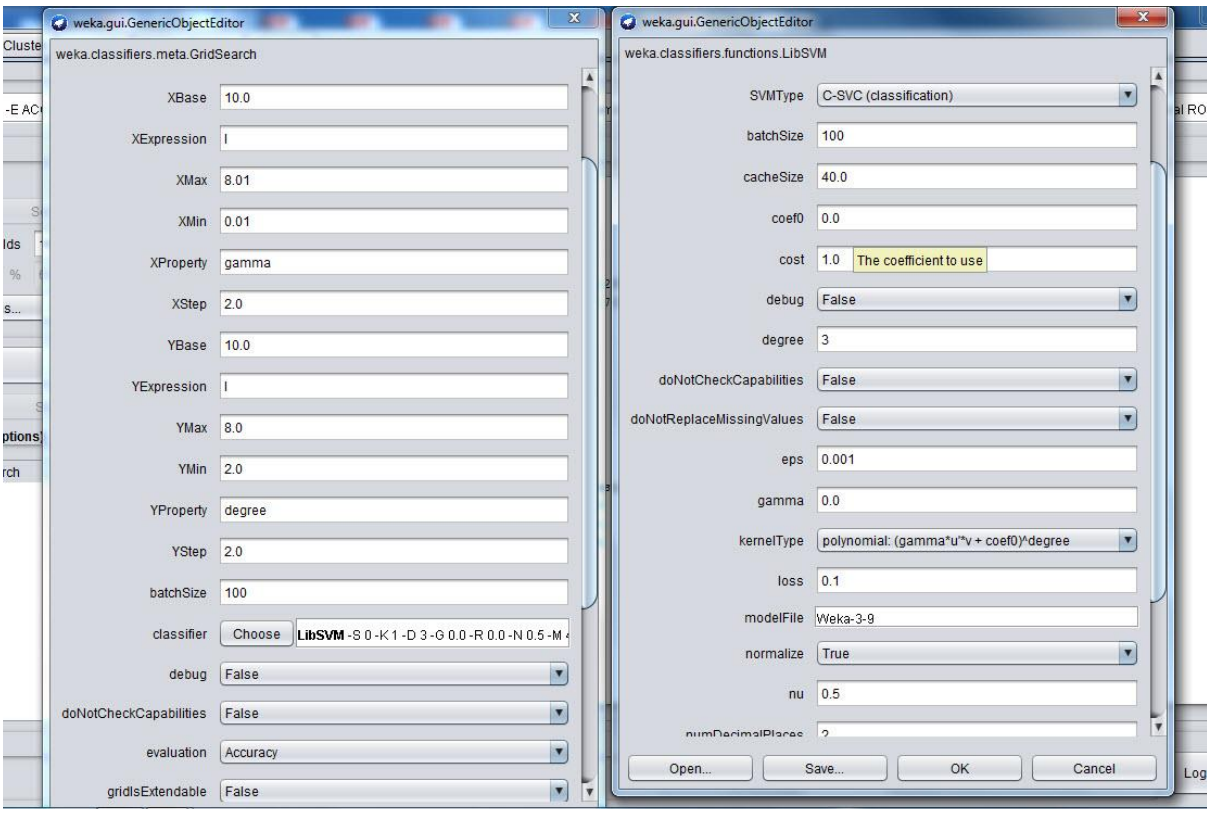


Screenshot: Results optimization using grid search for linear kernel

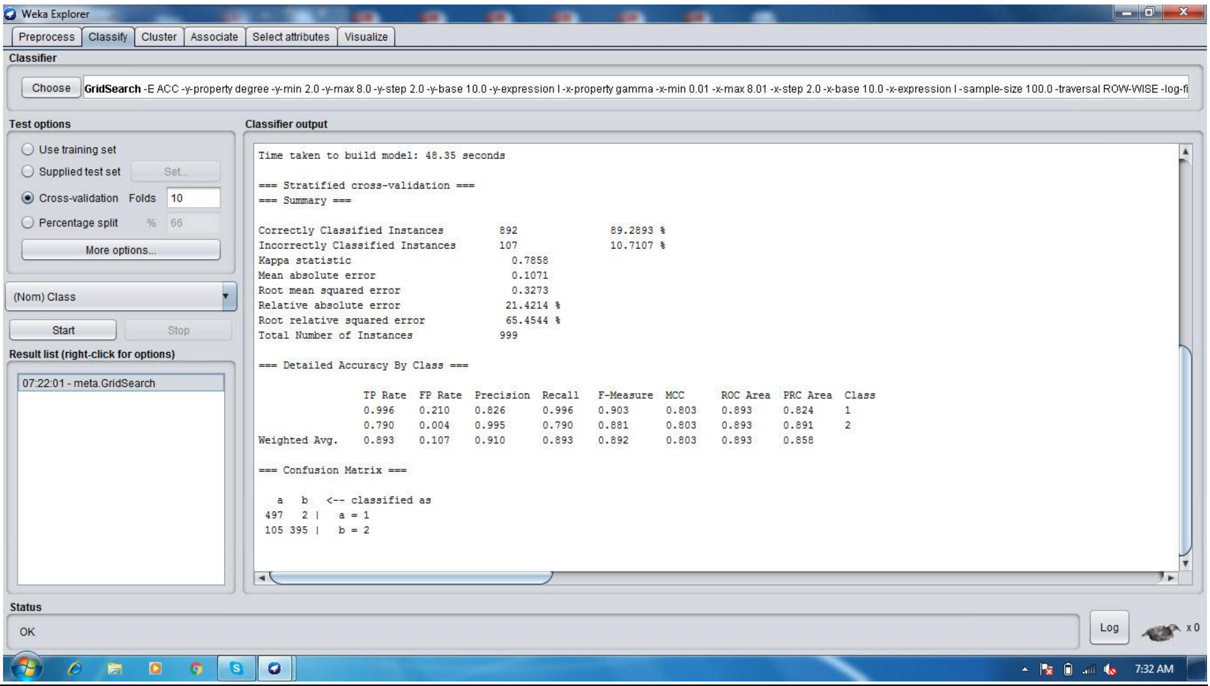


SVM –P (Polynomial kernel)

Screenshot: Tuning the parameters of SVM Polynomial Kernel using grid search

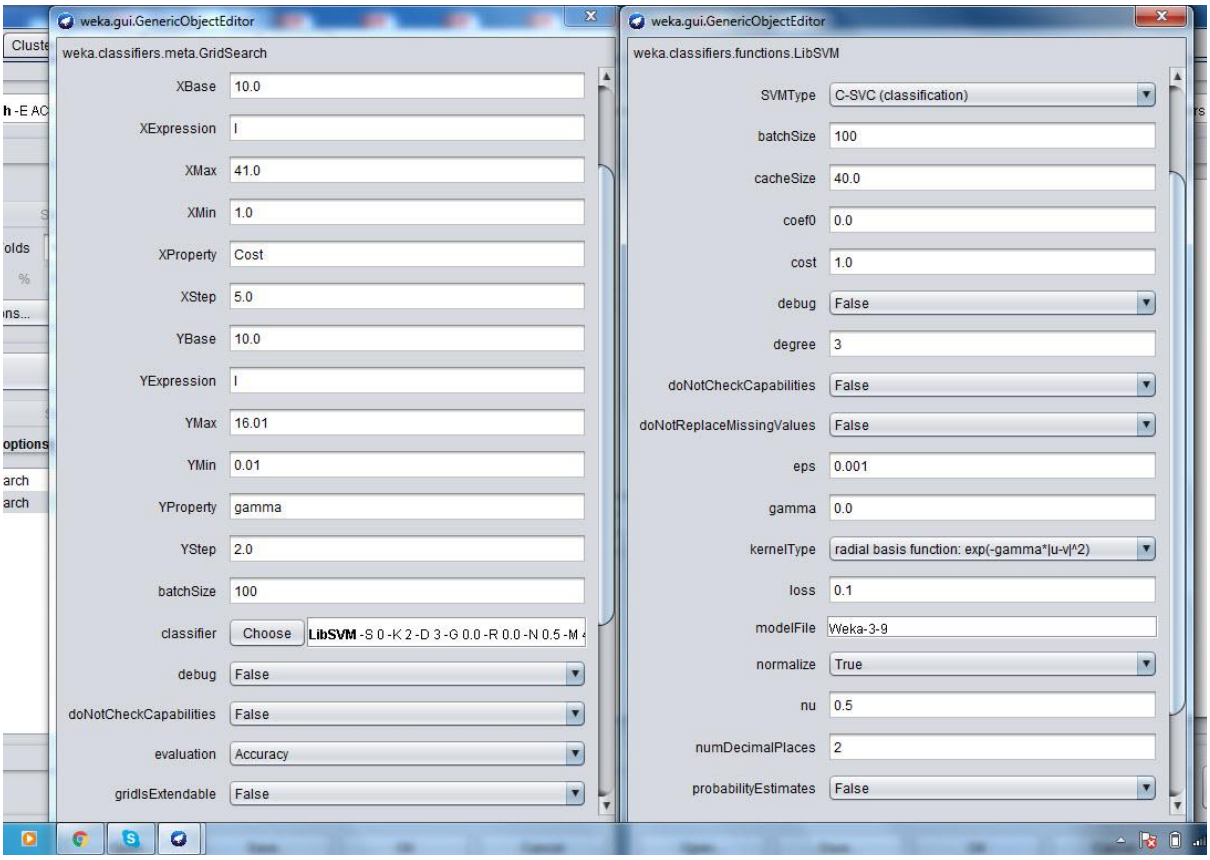


Screenshot: Results optimization using grid search for Polynomial kernel

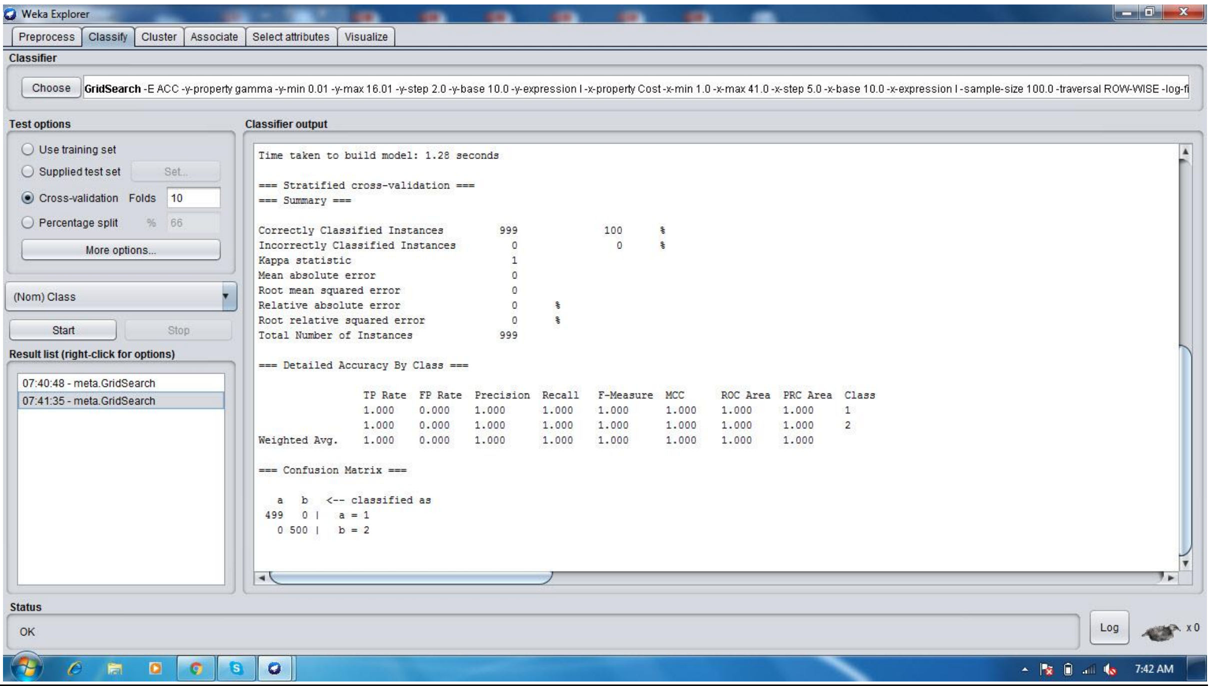


<u>SVM- R (Radial Base Function kernel)</u>

Screenshot: Tuning the parameters of SVM RBF Kernel using grid search



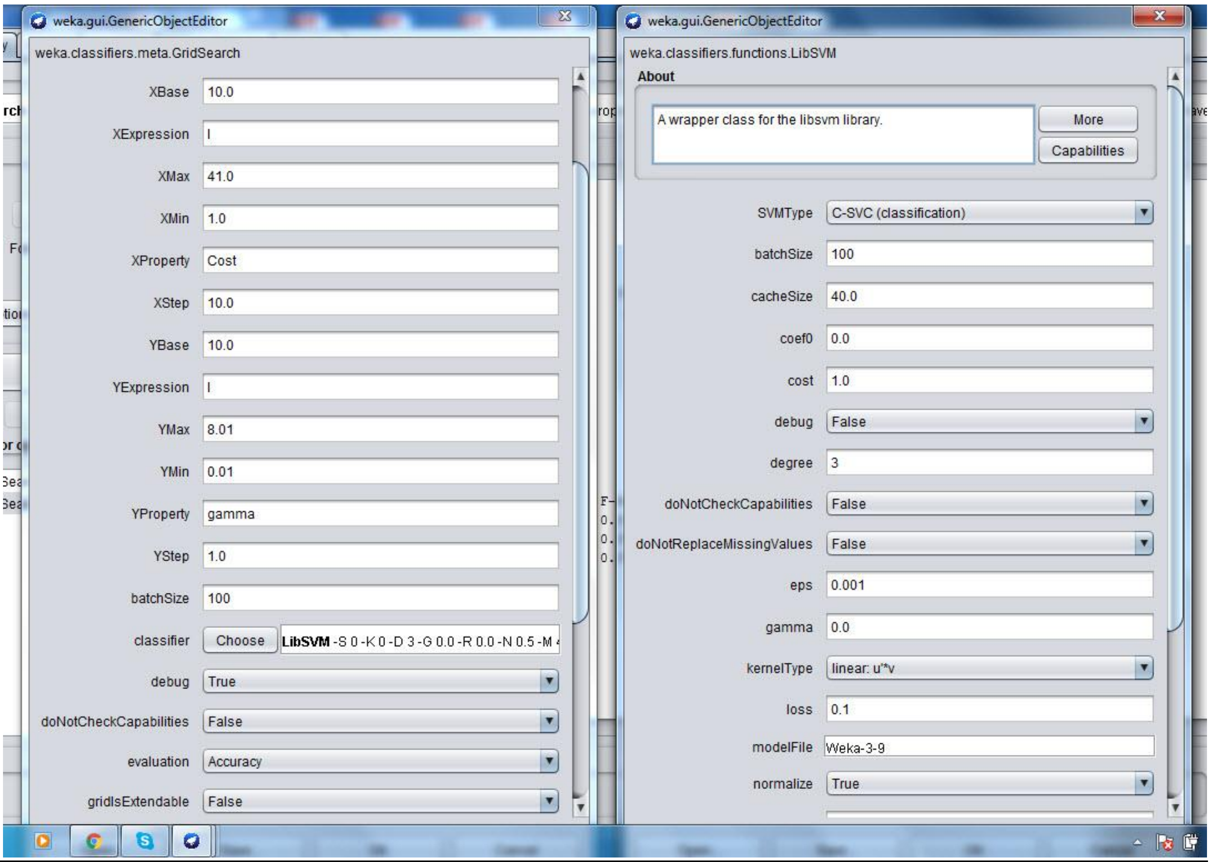
Screenshot: Results optimization using grid search for RBF kernel



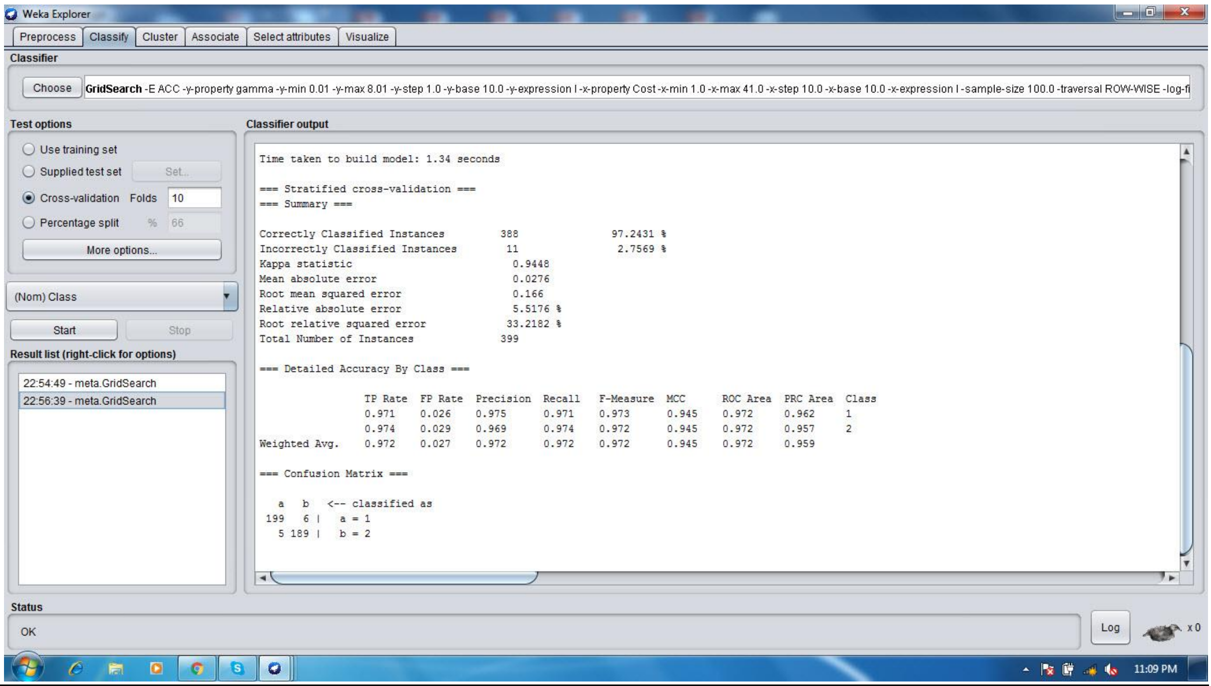
iii) TWO GAUSSIANS DATASET

SVM- L(Linear kernel)

Screenshot: Tuning the parameters of SVM Linear Kernel using grid search

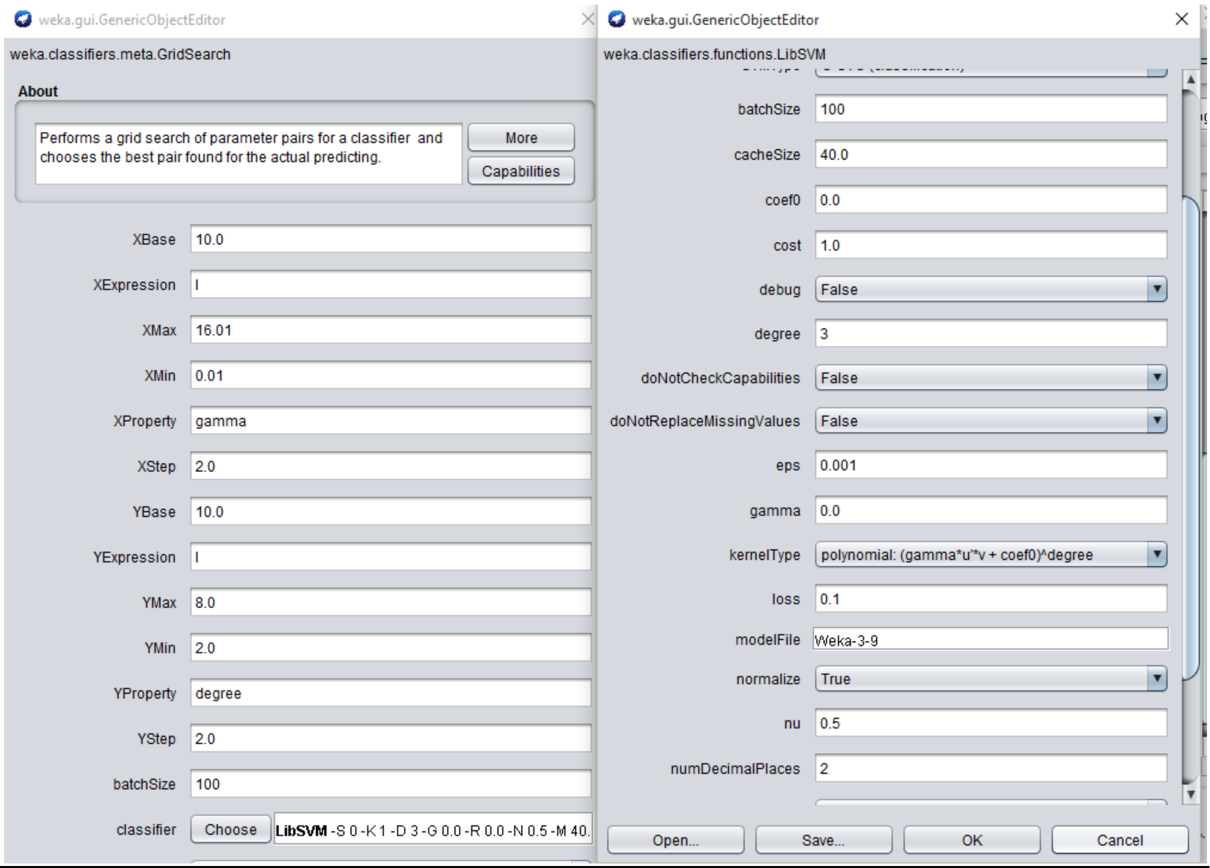


Screenshot: Results optimization using grid search for linear kernel

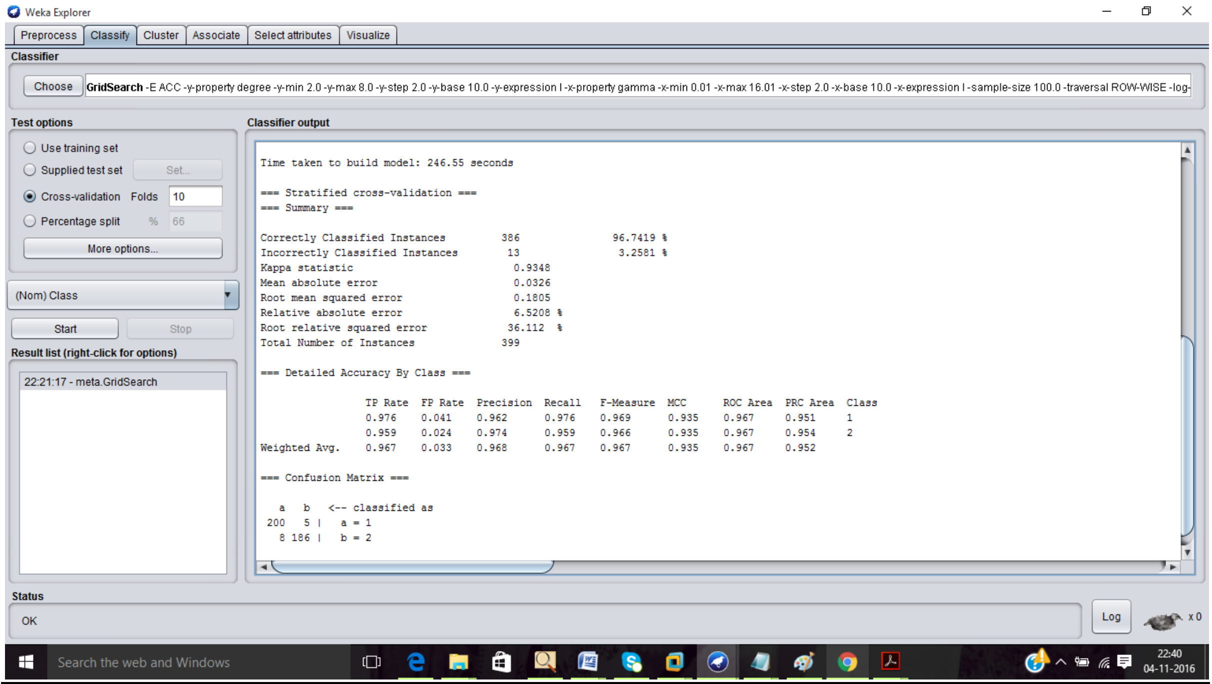


<u>SVM –P (Polynomial kernel)</u>

Screenshot: Tuning the parameters of SVM Polynomial Kernel using grid search

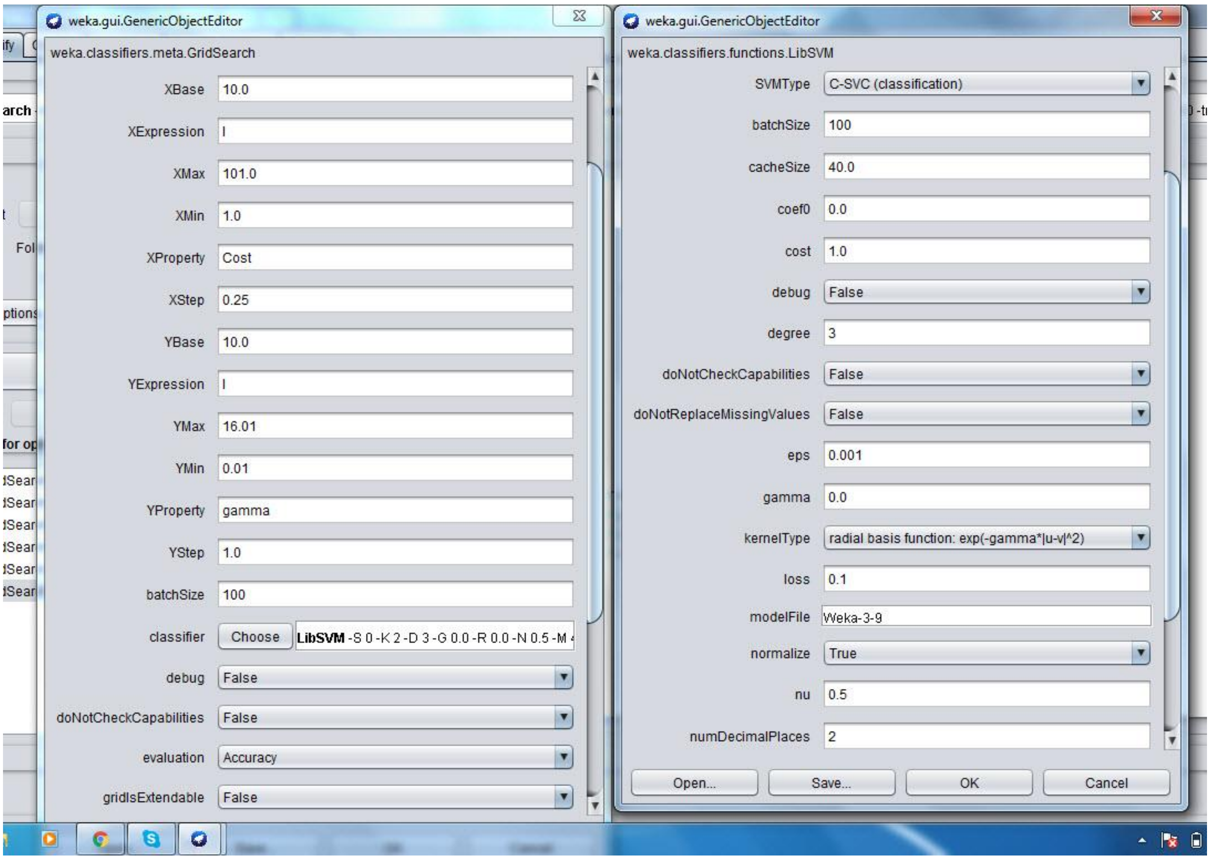


Screenshot: Results optimization using grid search for Polynomial kernel

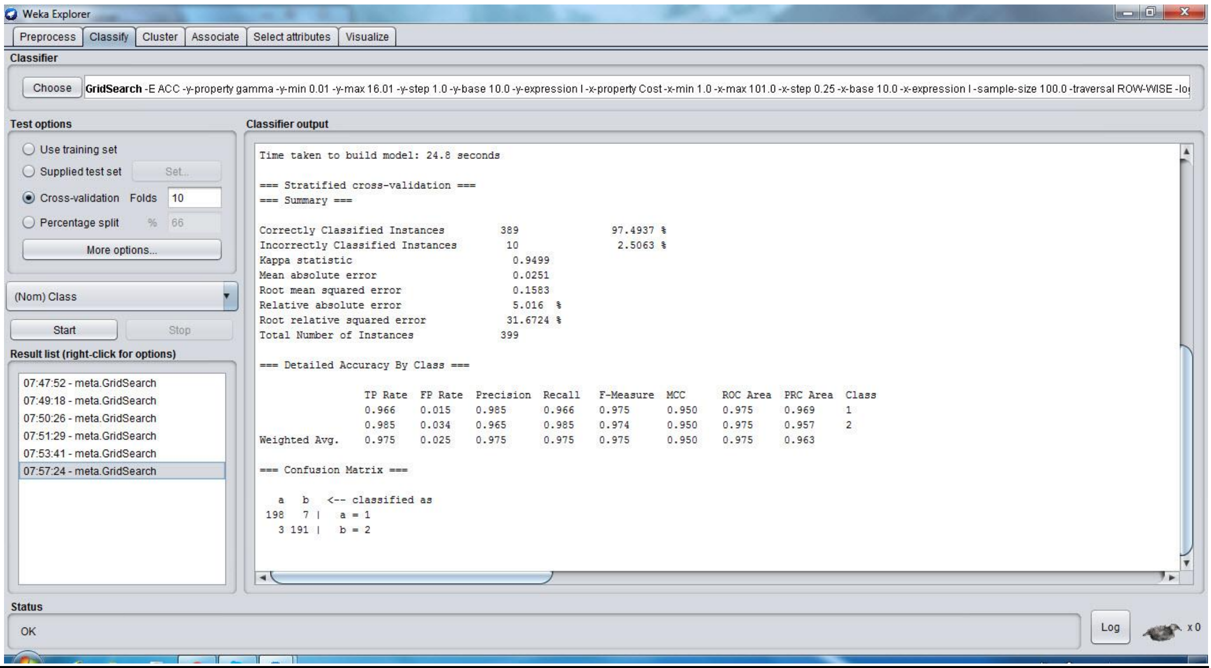


SVM- R (Radial Base Function kernel)

Screenshot: Tuning the parameters of SVM RBF Kernel using grid search



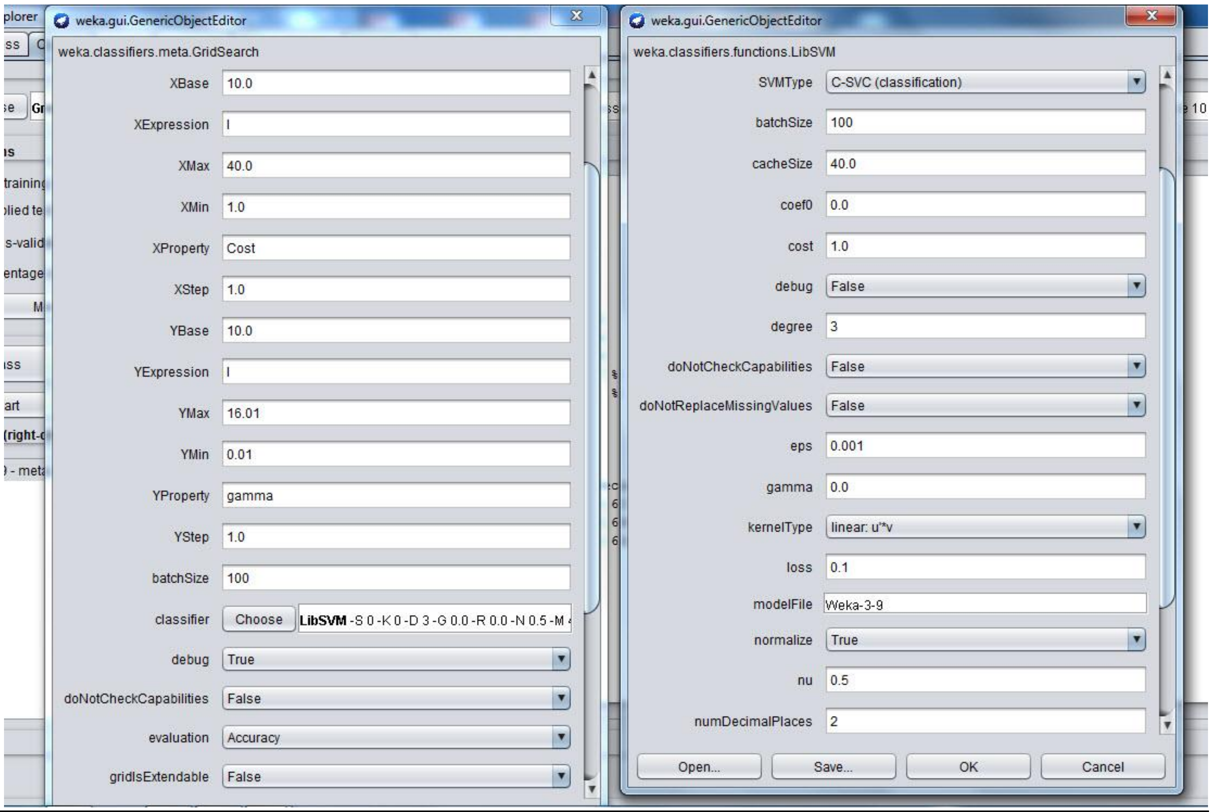
Screenshot: Results optimization using grid search for RBF kernel



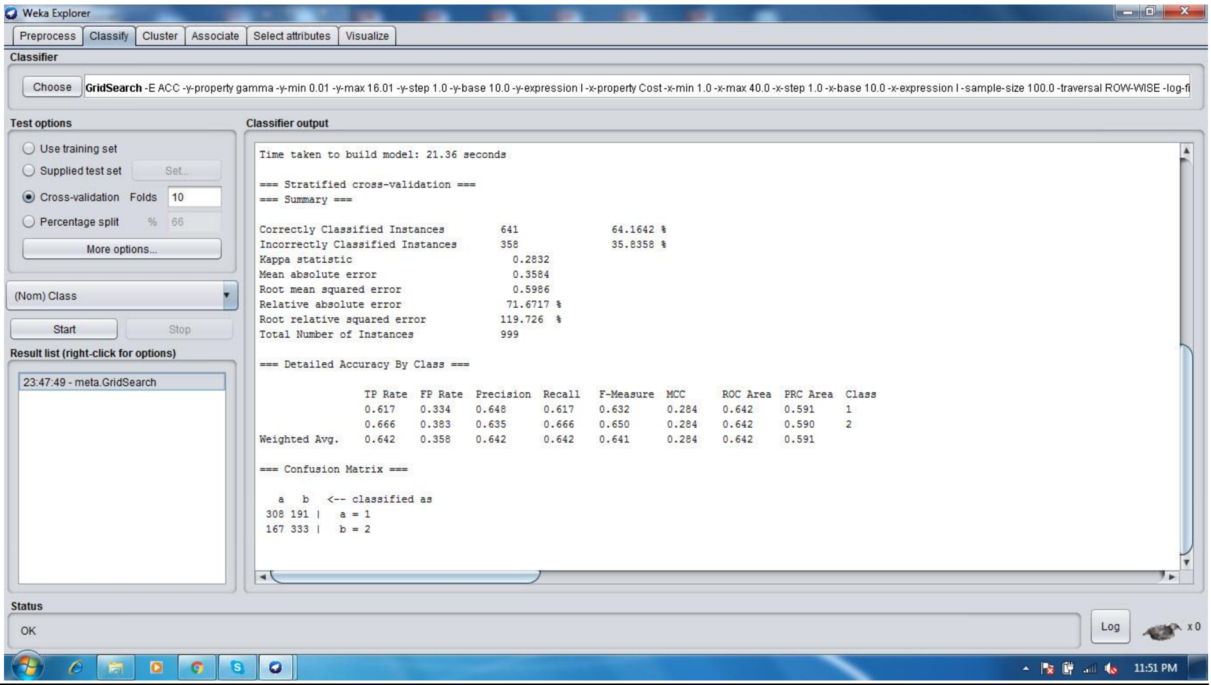
iv) TWO SPIRALS DATASET

<u>SVM- L(Linear kernel)</u>

<u>Screenshot: Tuning the parameters of SVM Linear Kernel using grid search</u>

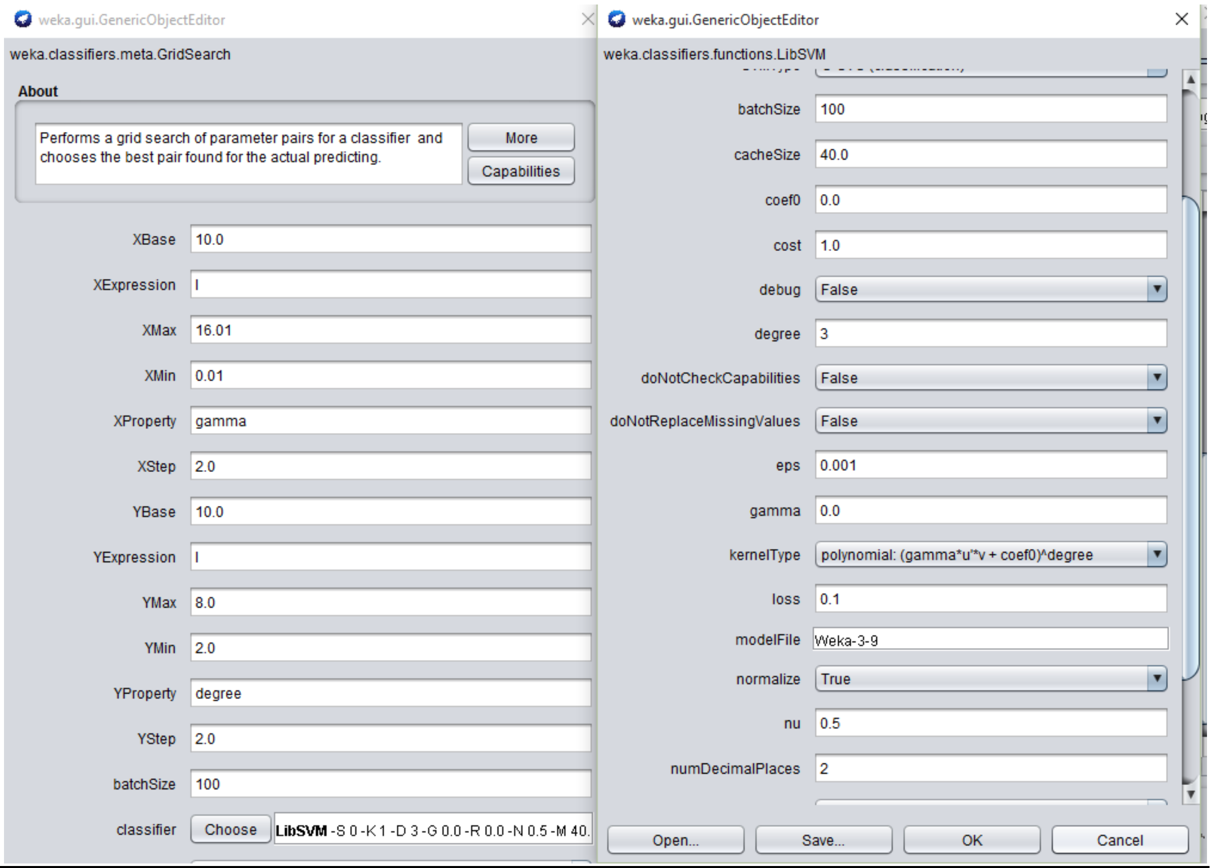


<u>Screenshot: Results optimization using grid search for linear kernel</u>

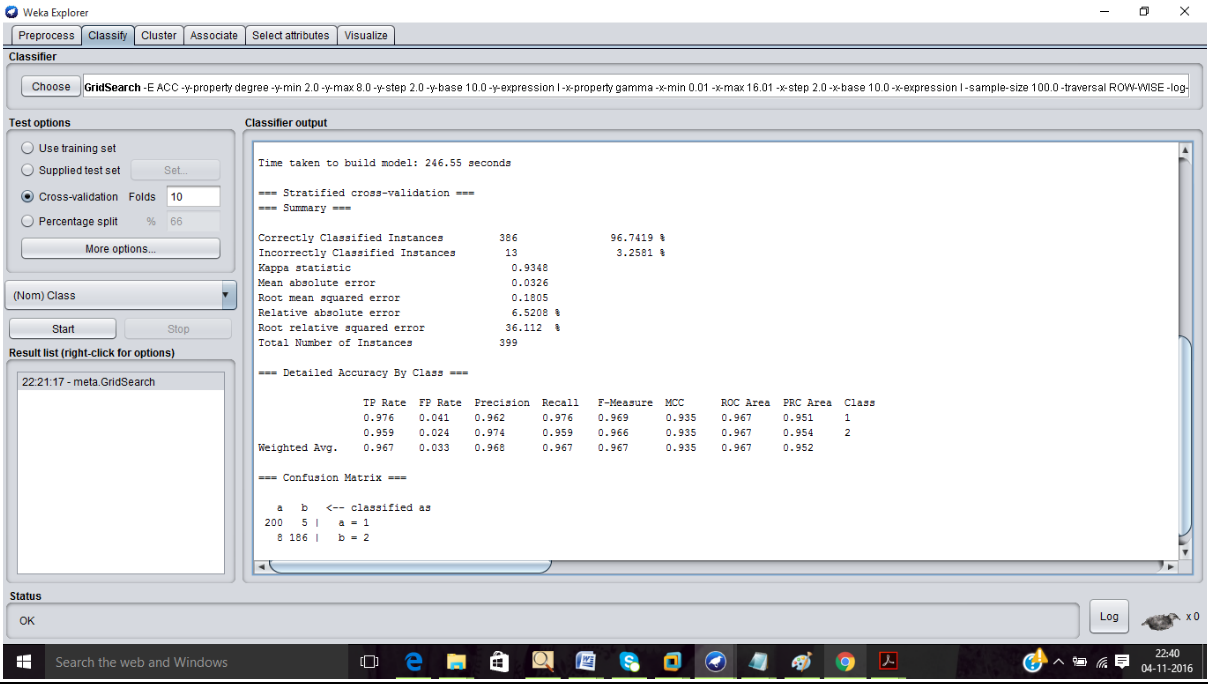


SVM –P (Polynomial kernel)

Screenshot: Tuning the parameters of SVM Polynomial Kernel using grid search

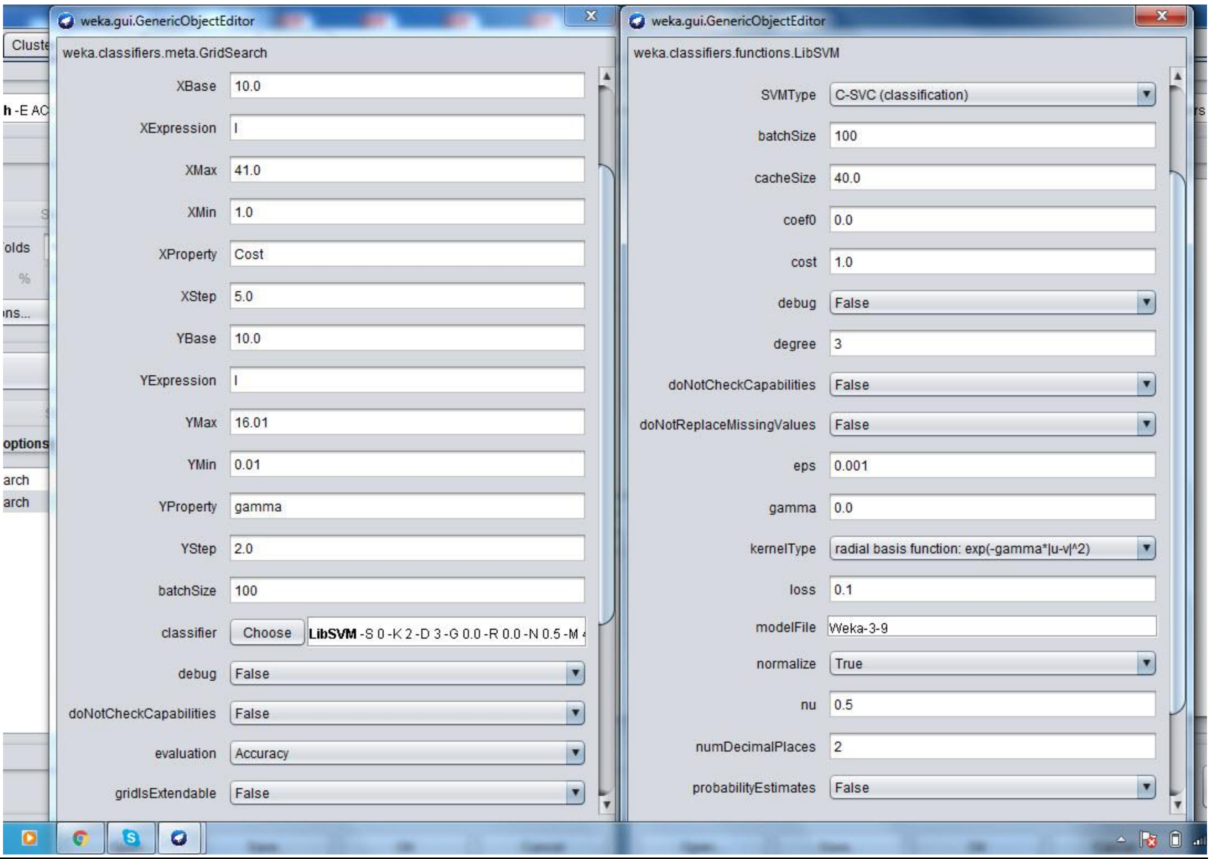


Screenshot: Results optimization using grid search for Polynomial kernel

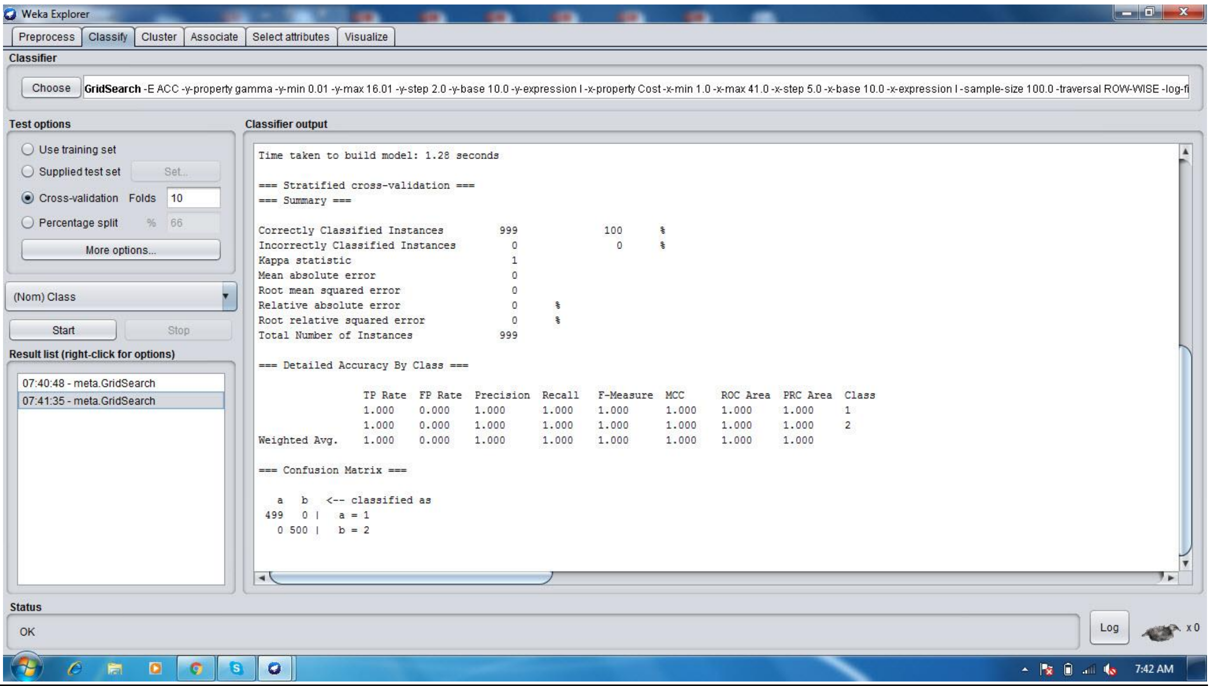


<u>SVM- R (Radial Base Function kernel)</u>

Screenshot: Tuning the parameters of SVM RBF Kernel using grid search



Screenshot: Results optimization using grid search for RBF kernel



5) **Comparison of SVM Classifiers between Grid Search and Default Parameters**

Table: Comparison of SVM Kernels Accuracy (Performance Metrics) for various datasets by using default parameters and grid optimization techniques.

Classifiers	**SVM Default Parameters**			**SVM Grid Search Optimisation**		
Datasets	SVM-L	SVM-P	SVM-R	SVM-L	SVM-P	SVM-R
Cluster In Cluster	53.4535%	62.2623%	100%	53.5536%	88.0881%	100%
Half Kernel	73.5736%	63.4635%	100%	73.6737%	89.2893%	100%
Two Gaussians	97.2431%	96.9925%	97.2431%	97.3431%	96.7419%	97.4937%
Two Spirals	64.0641%	99.9499%	95.1952%	64.1642%	96.8419%	95.2952%

The choice of kernel depends on the problem and is data dependent. The Euclidean geometry can be altered according to our problem. The choice can be automated by introducing cross validation based model selection Proper kernel has to be chosen for SVM to perform better.

- When the number of features is large, SVM linear kernel is preferable. Linea
- SVM linear is the special case of SVM RBF. The computation is faster for linear and it can be classified using LibLinear function as well.
- The polynomial kernel serves good when the classification is non linear in lower dimensional space and linear in higher dimensional space. Sometimes, through kernel trick the solution can be obtained without explicitly mapping it to higher dimension. As the degree of the polynomial increases, the size of the function class also increases. In some cases, it becomes powerful as it may lead to infinite order polynomial kernel.
- However, Radial basis function kernel (Gaussian kernel) can be used when there are a number of observations.
- RBF is non parametric whereas polynomial is parametric. The complexity of the model grows upto infinite when it is non parametric and it can establish complex relationships between

data. In contrast, if it is parameterized, the model size is fixed hence data is saturated and weak assumptions.

- Grid search is used to optimize the parameters and tune the SVM kernels by classifying them with better performance metrics. Grid search varies the parameters of Cost (C), gamma and degree (d). The SVM kernels is classified better with grid search optimization than using default parameters for classification.

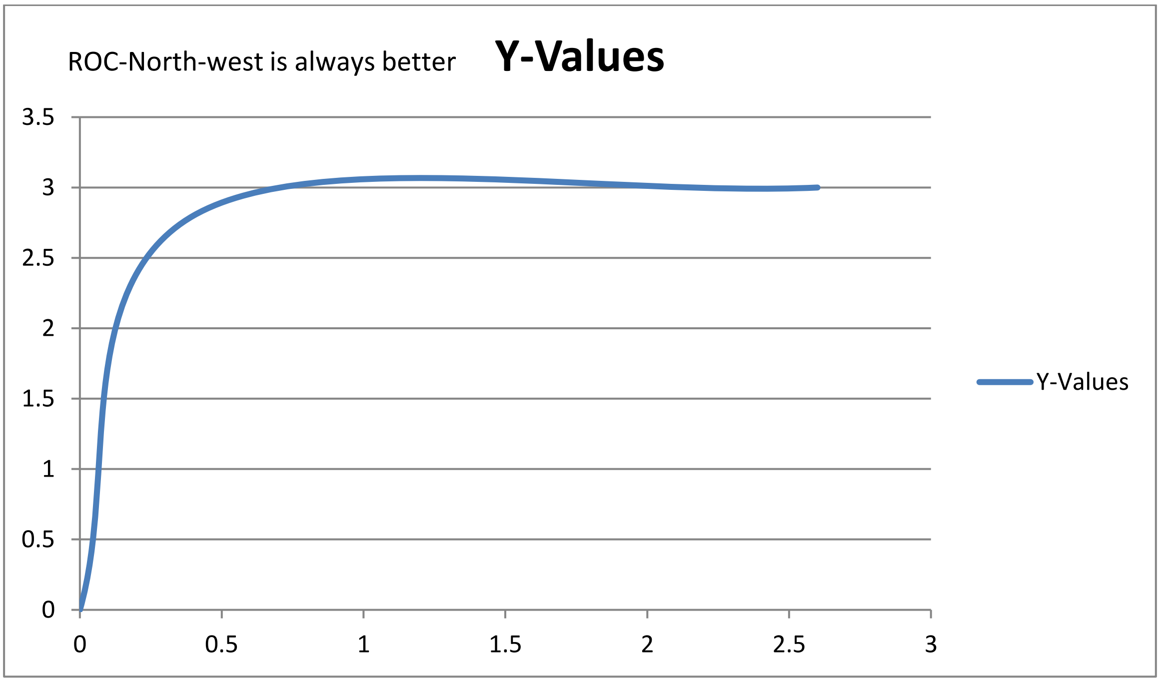
6) **ROC Curve and AUC for each dataset**

An ROC graph is actually two-dimensional graph in which True Positive Rate (TPR) is plotted on the Y axis and False Positive Rate (FPR) is plotted on the X axis. ROC performance of a classifier is usually represented by a value which is the area under the ROC curve (AUC).

In Weka, after running the classifier for each dataset, choose 'Visualize threshold curve' and select 1 or 2 based on the class.

The points obtained are connected using cut off levels, which forms the convex skull. ROC (Receiver Operating Characteristic) graph serves as a tool for analysing the tradeoffs between TP (Sensitivity) and FP rates (1-Specificity) and identifying misclassification. However, it is more preferred for unbalanced class problems. One point corresponds to the result of a classifier.

If the points lie in the North West region it gives more accuracy and is always better. On the other hand, points lying in South East are worse. The points along the diagonals x=y denotes a random classifier. The curve in the ROC known as a ROC curve varies based on the parameters and samples in practice, while in theory, it will be an exact curve.

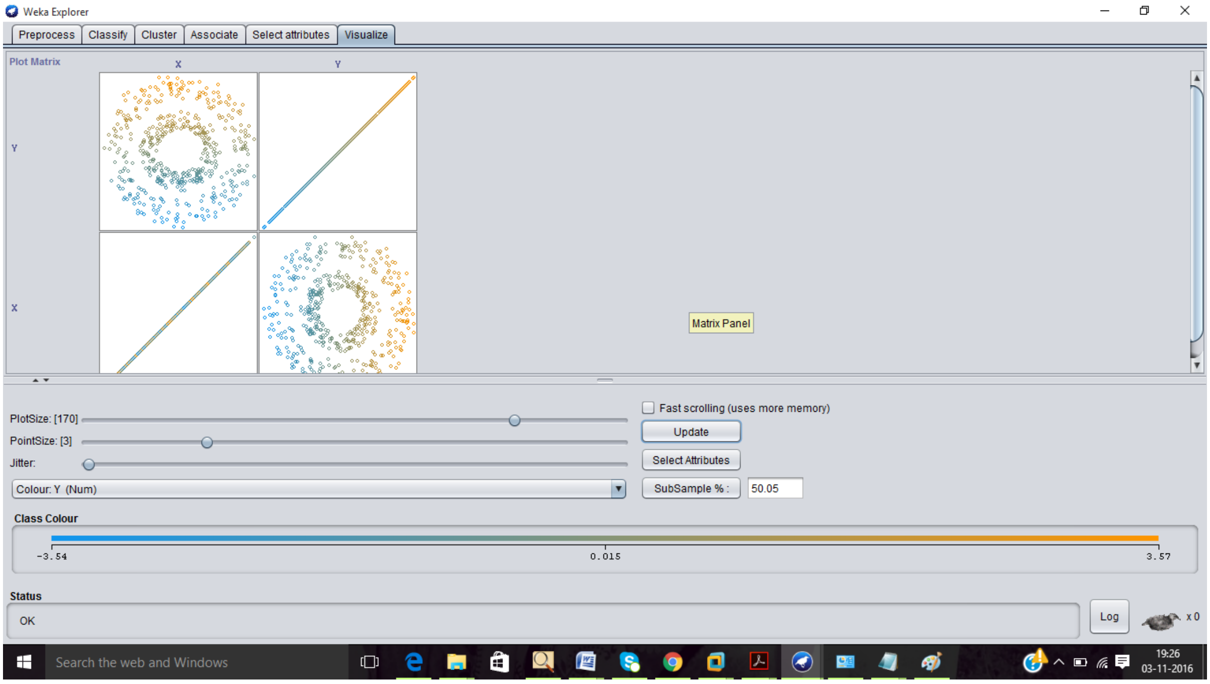


AUC – AREA UNDER THE ROC CURVE

It is the region under the ROC curve to give a quantitative comparison. It has computed through Trapezoidal approximation, curve fitting and integration. Different LDR methods have used for coupling linear and quadratic classifiers. The value of AUC is between 0 and 1.

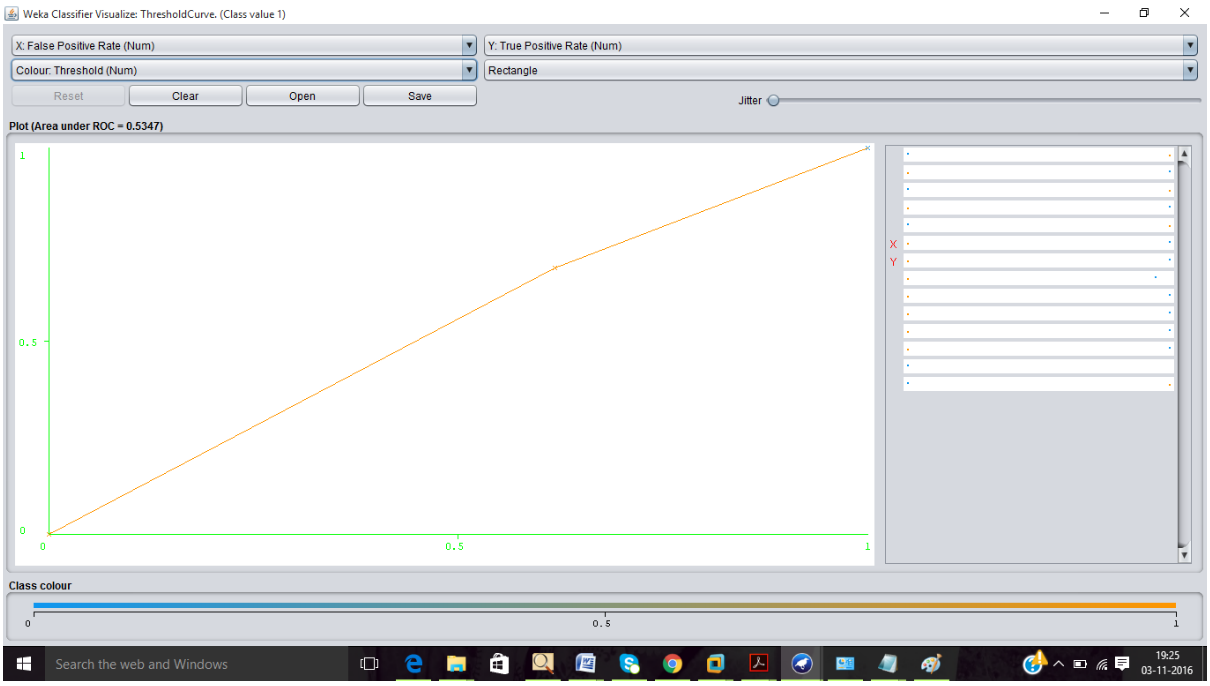
i) CLUSTER IN CLUSTER DATASET

Screenshot: Classes 1 and 2 samples are visualised in red and blue colour respectively

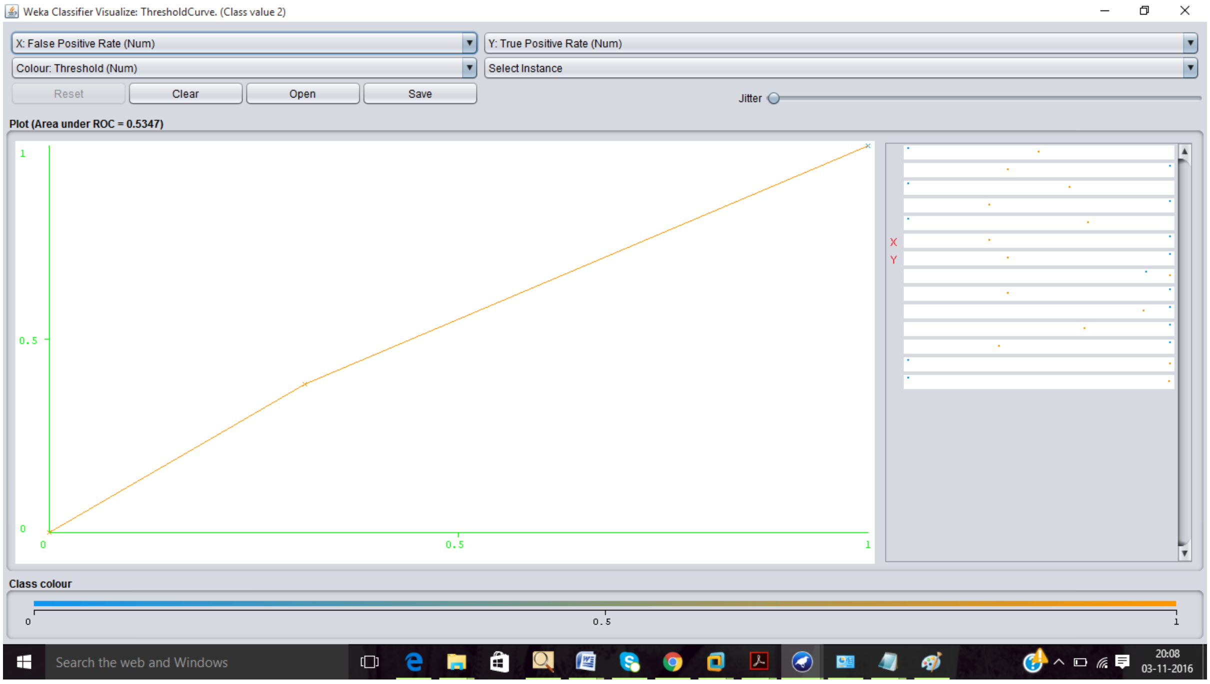


<u>SVM- L (Linear kernel)</u>

Screenshot: For SVM-Linear, Visualizing the threshold curve for Class 1

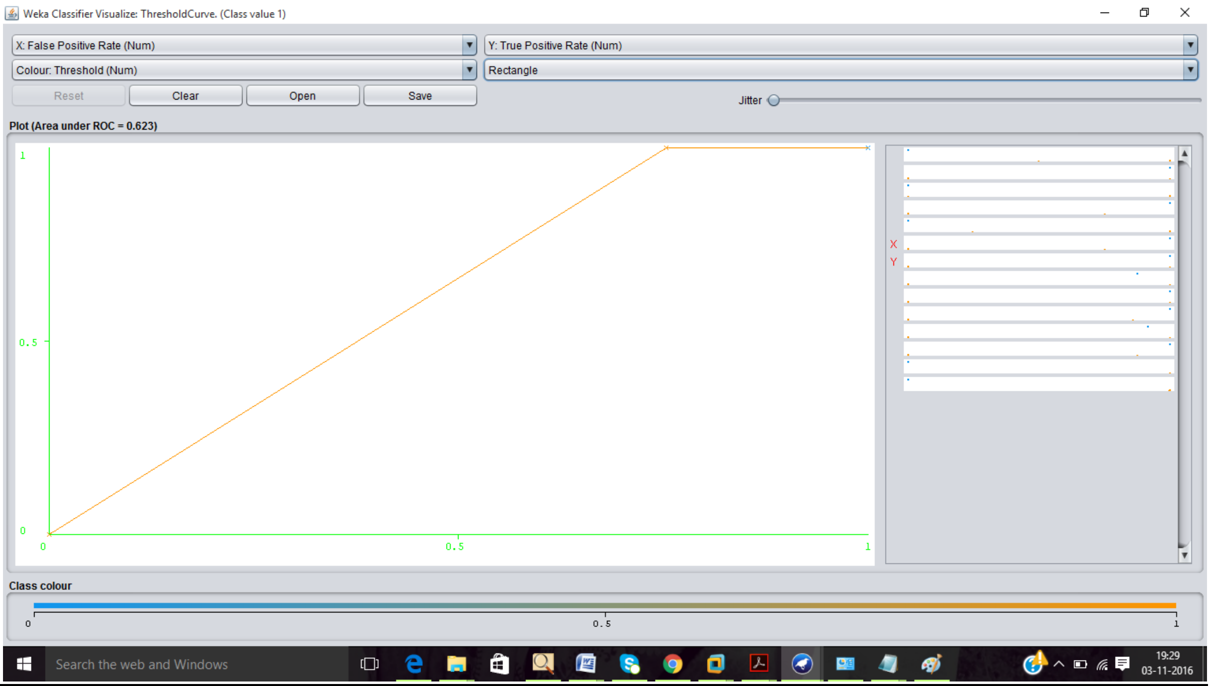


Screenshot: For SVM-Linear, Visualizing the threshold curve for Class 2

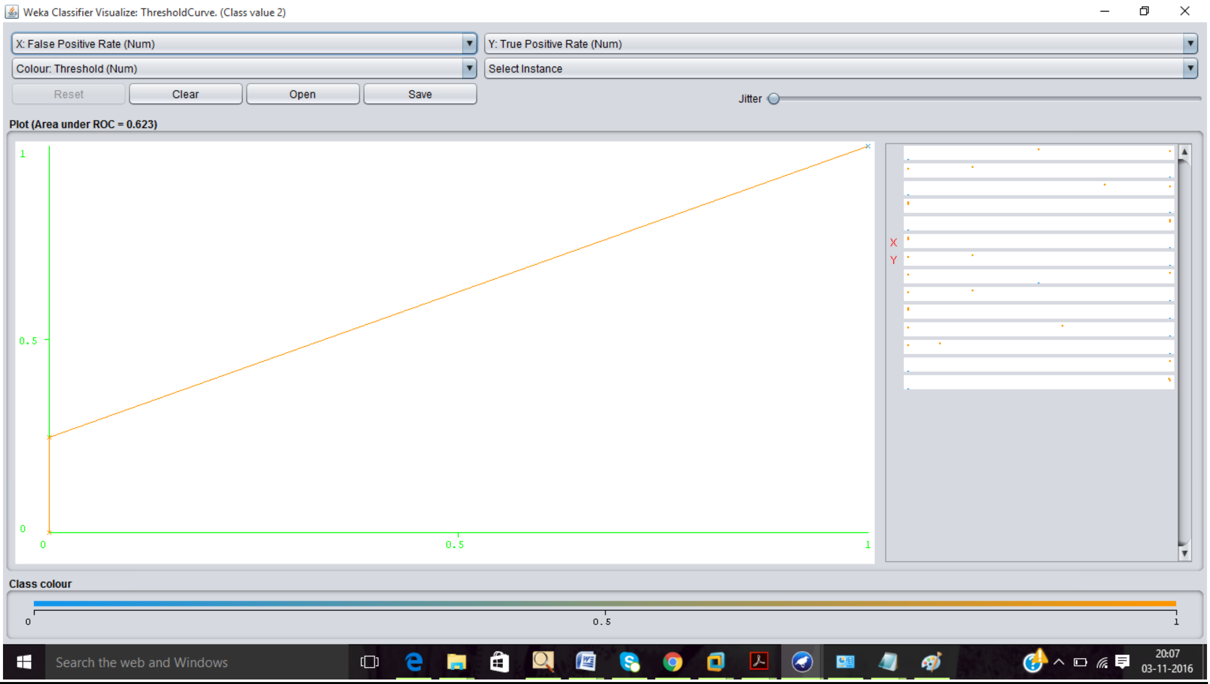


SVM –P (Polynomial kernel)

Screenshot: For SVM-Polynomial, Visualizing the threshold curve for Class 1

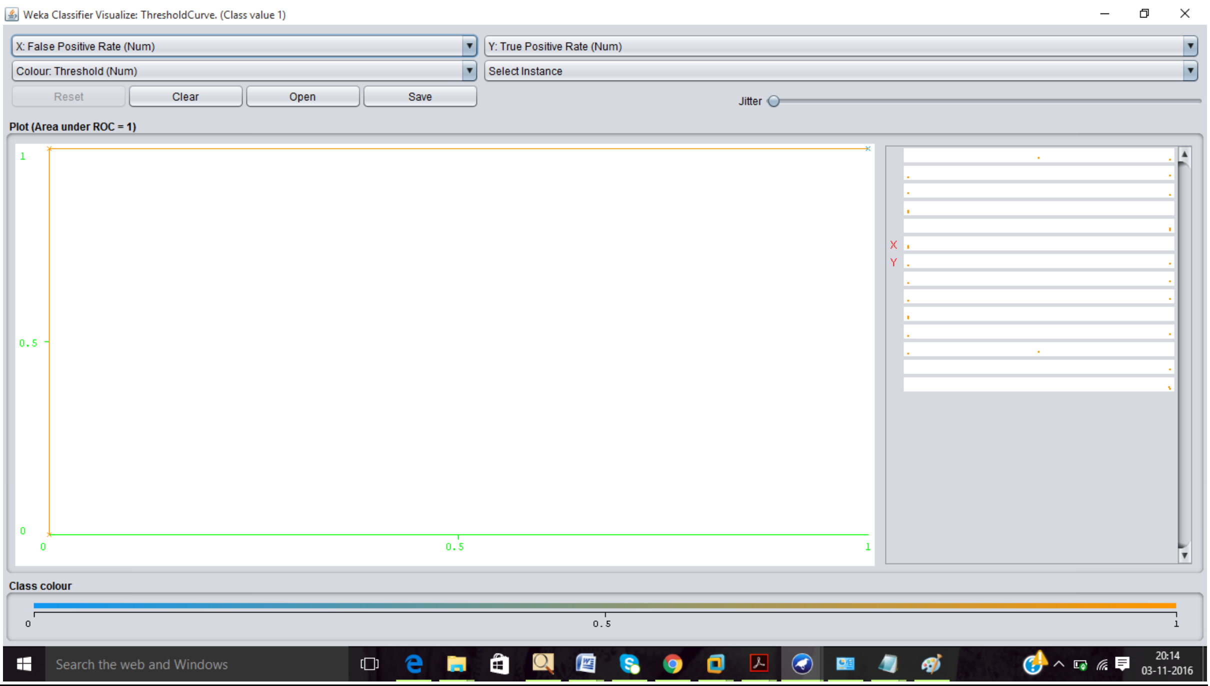


Screenshot: For SVM-Polynomial, Visualizing the threshold curve for Class 2

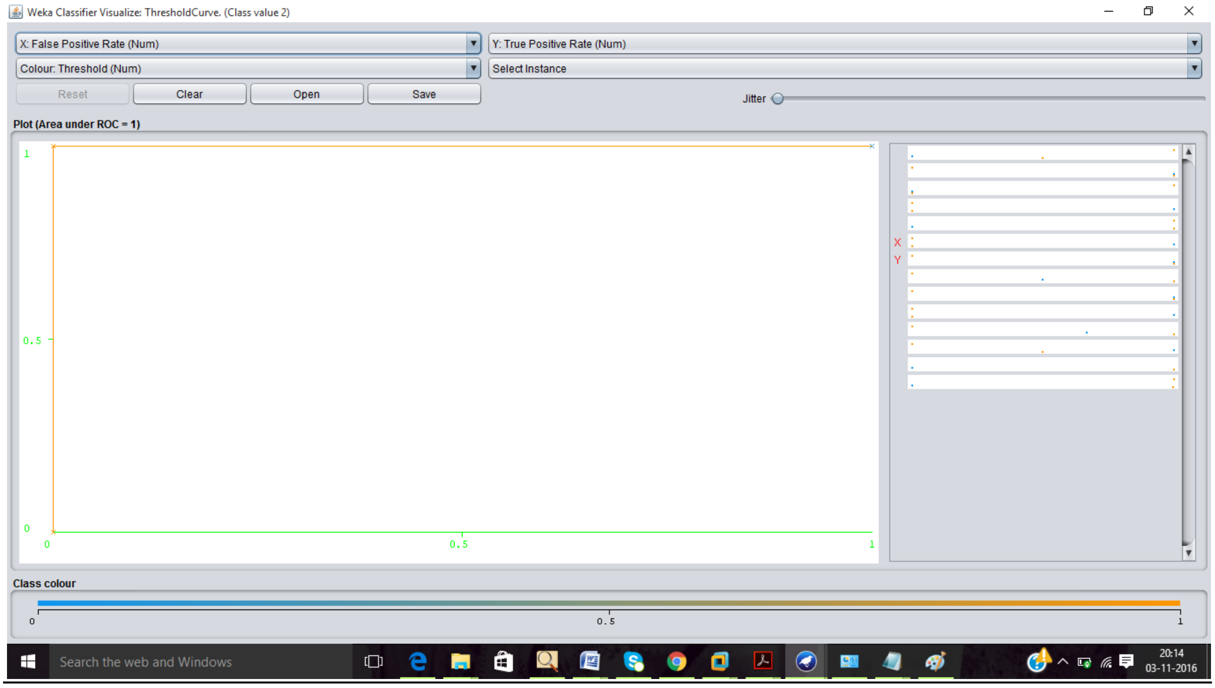


SVM- R (Radial Base Function kernel)

Screenshot: For SVM-RBF, Visualizing the threshold curve for Class 1

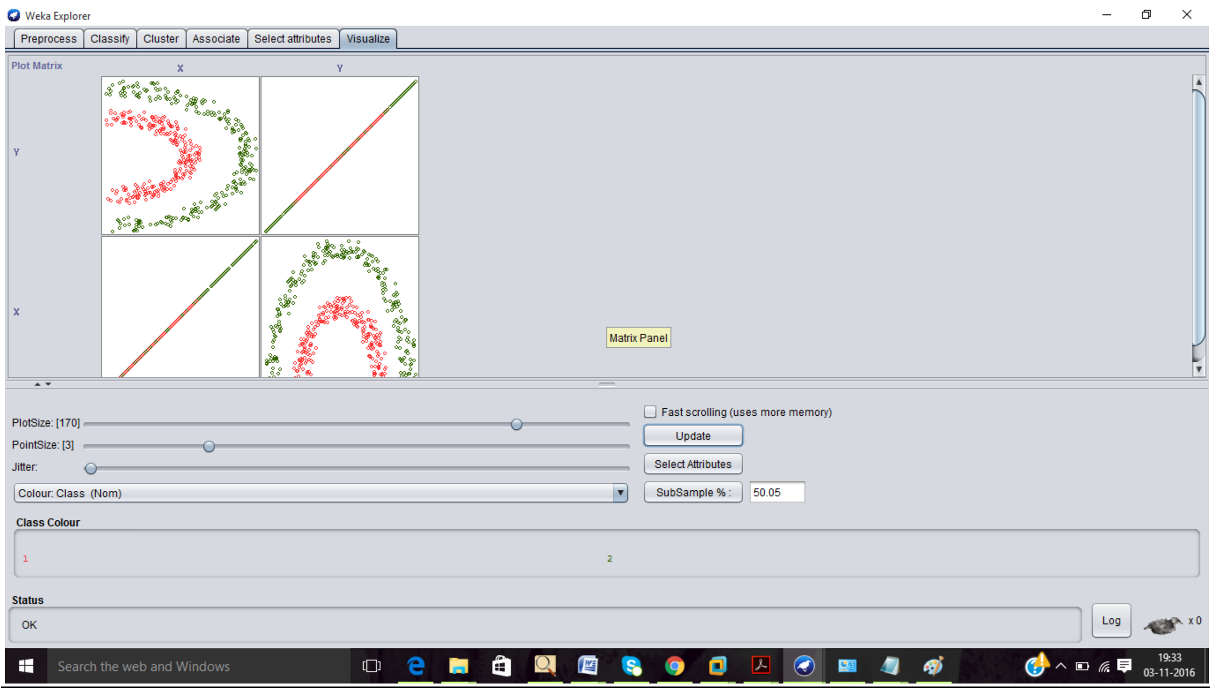


Screenshot: For SVM-RBF, Visualizing the threshold curve for Class 2



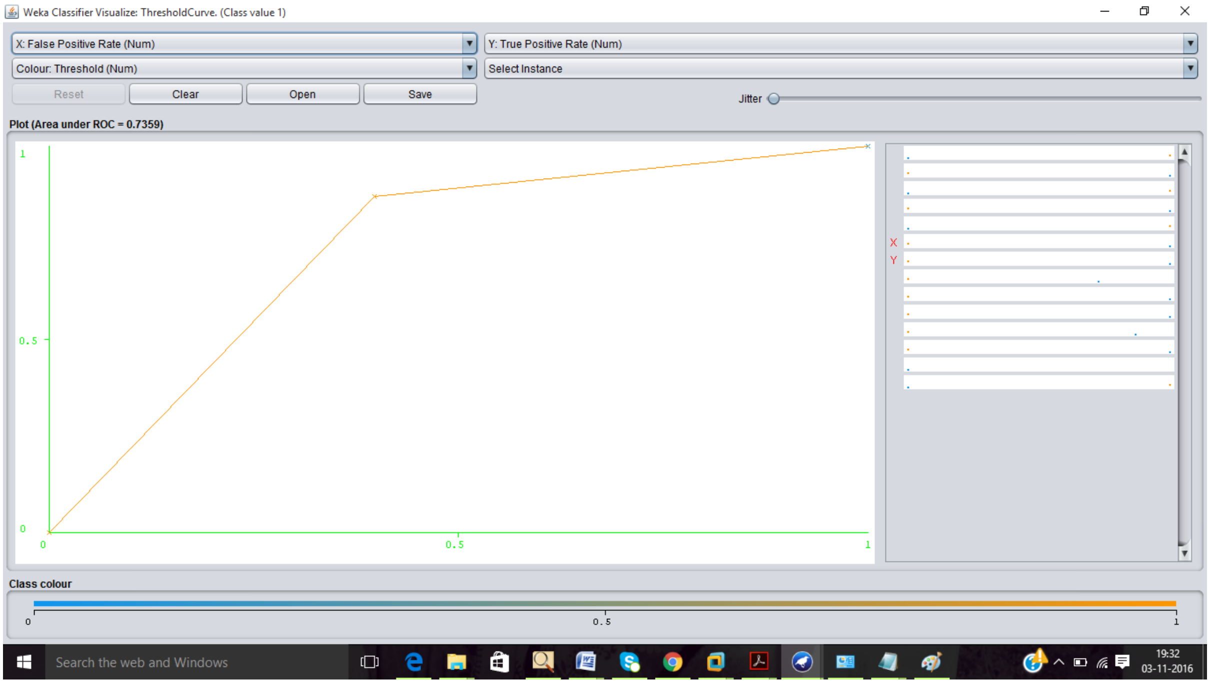
ii) HALF KERNAL DATASET

Screenshot: Classes 1 and 2 samples are visualised in red and green colour respectively

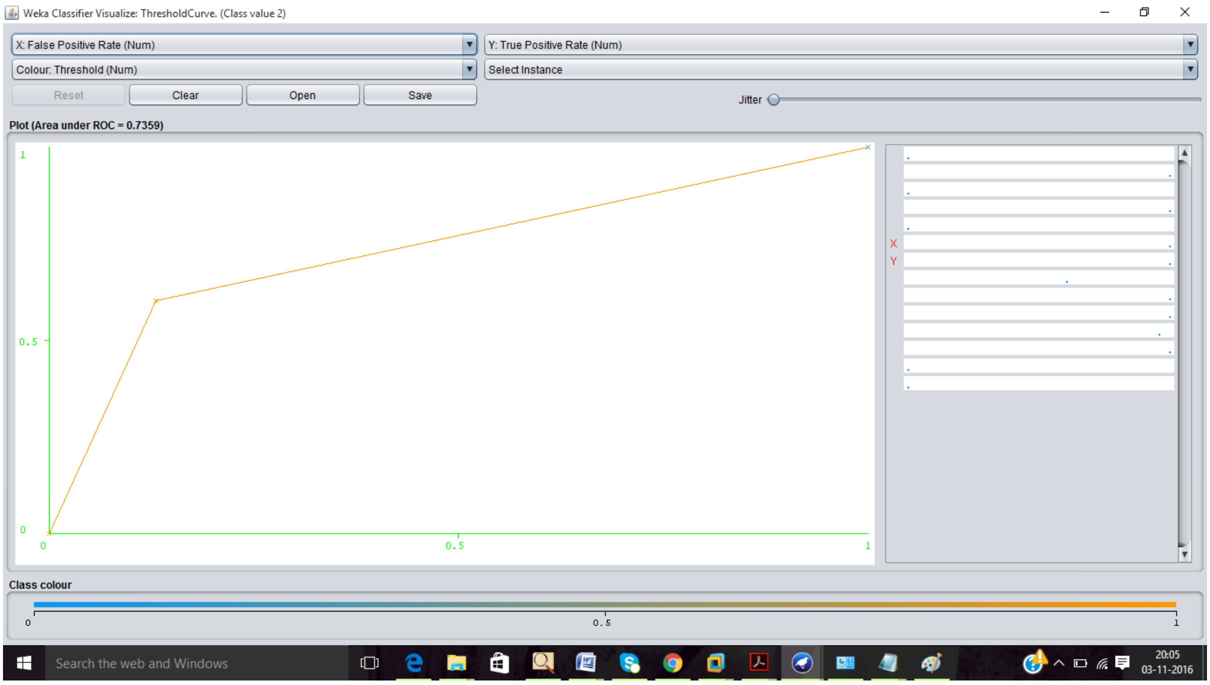


<u>SVM- L(Linear kernel)</u>

Screenshot: For SVM-Linear, Visualizing the threshold curve for Class 1

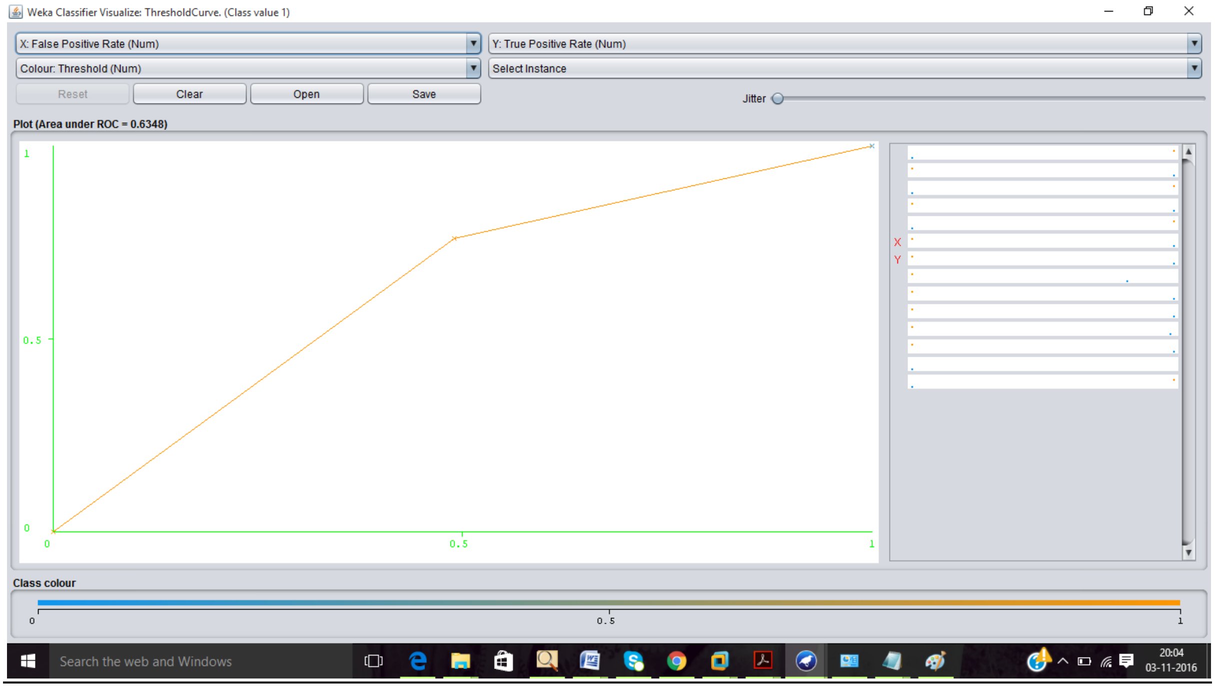


Screenshot: For SVM-Linear, Visualizing the threshold curve for Class 2

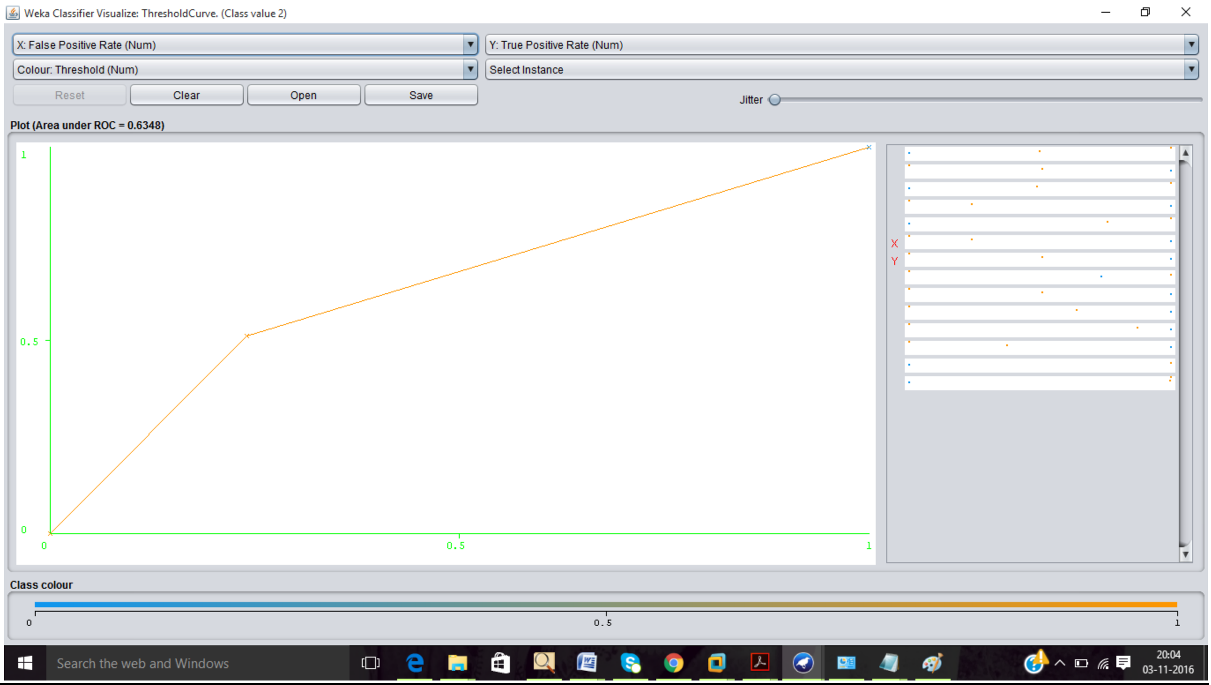


<u>SVM –P (Polynomial kernel)</u>

Screenshot: For SVM-Polynomial, Visualizing the threshold curve for Class 1

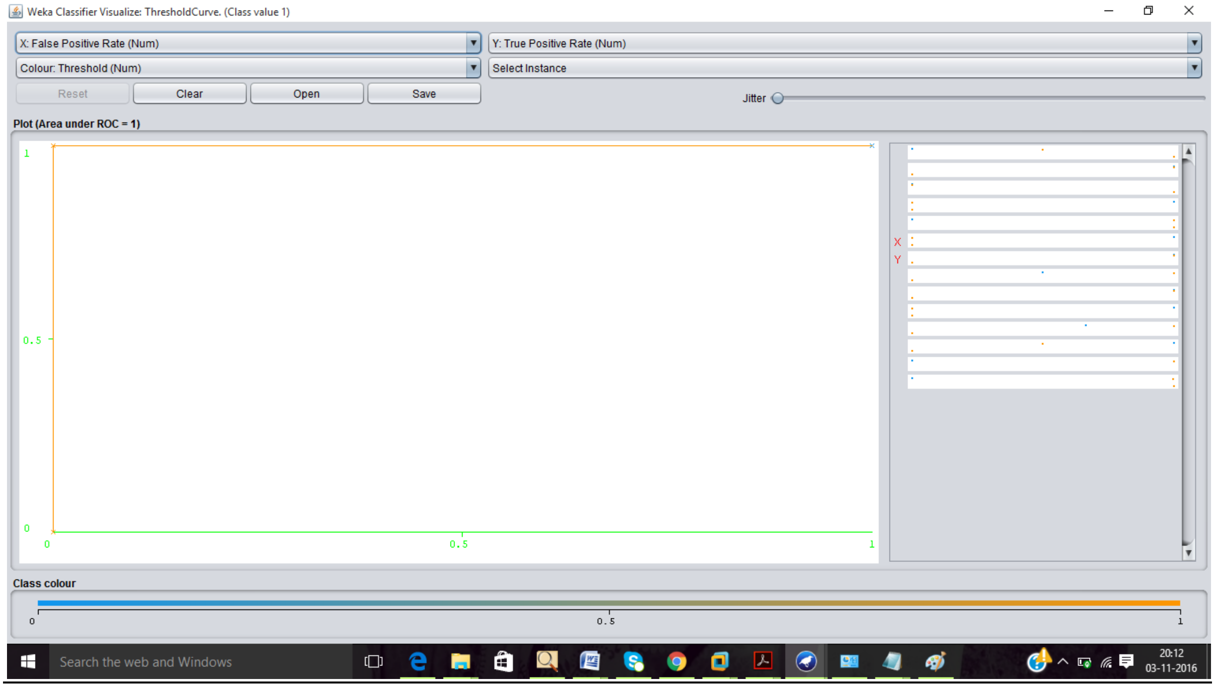


Screenshot: For SVM-Polynomial, Visualizing the threshold curve for Class 2

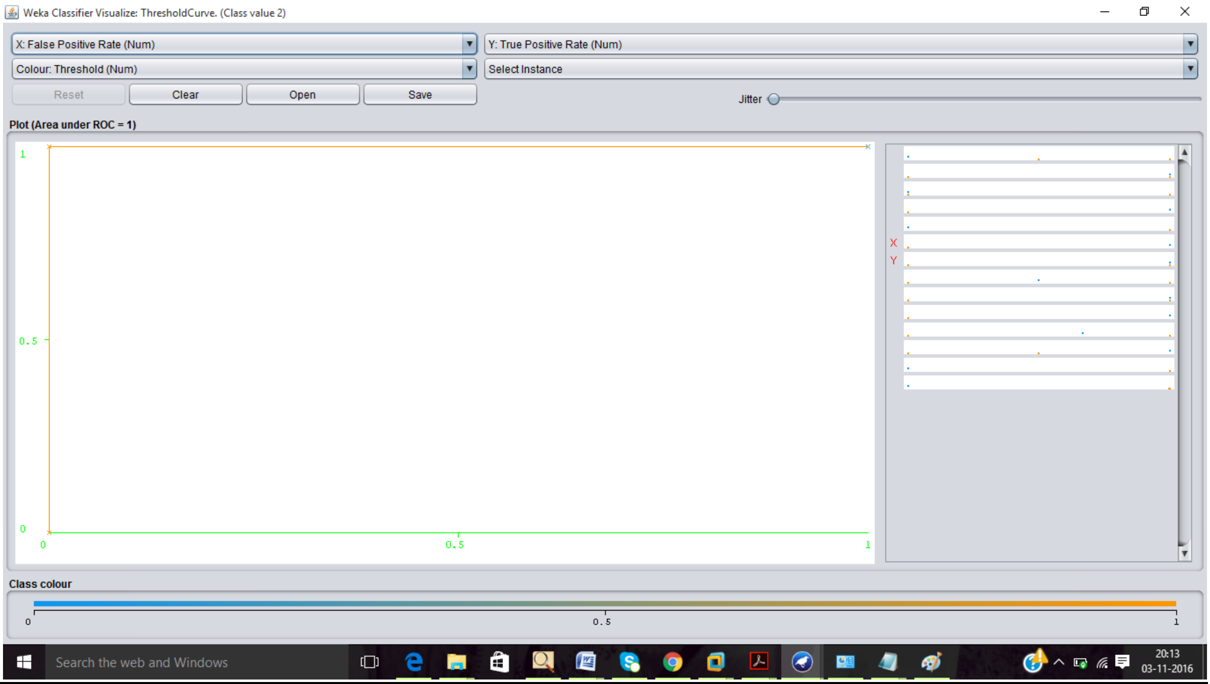


<u>SVM- R (Radial Base Function kernel)</u>

Screenshot: For SVM-RBF, Visualizing the threshold curve for Class 1

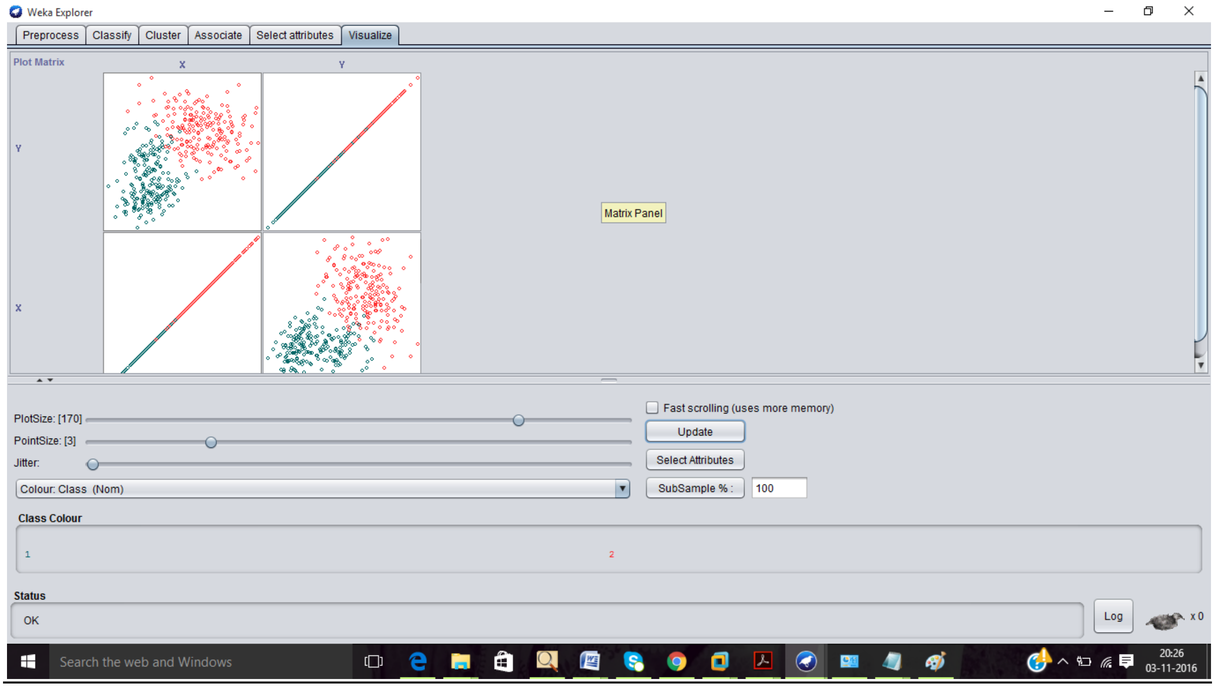


Screenshot: For SVM-RBF, Visualizing the threshold curve for Class 2



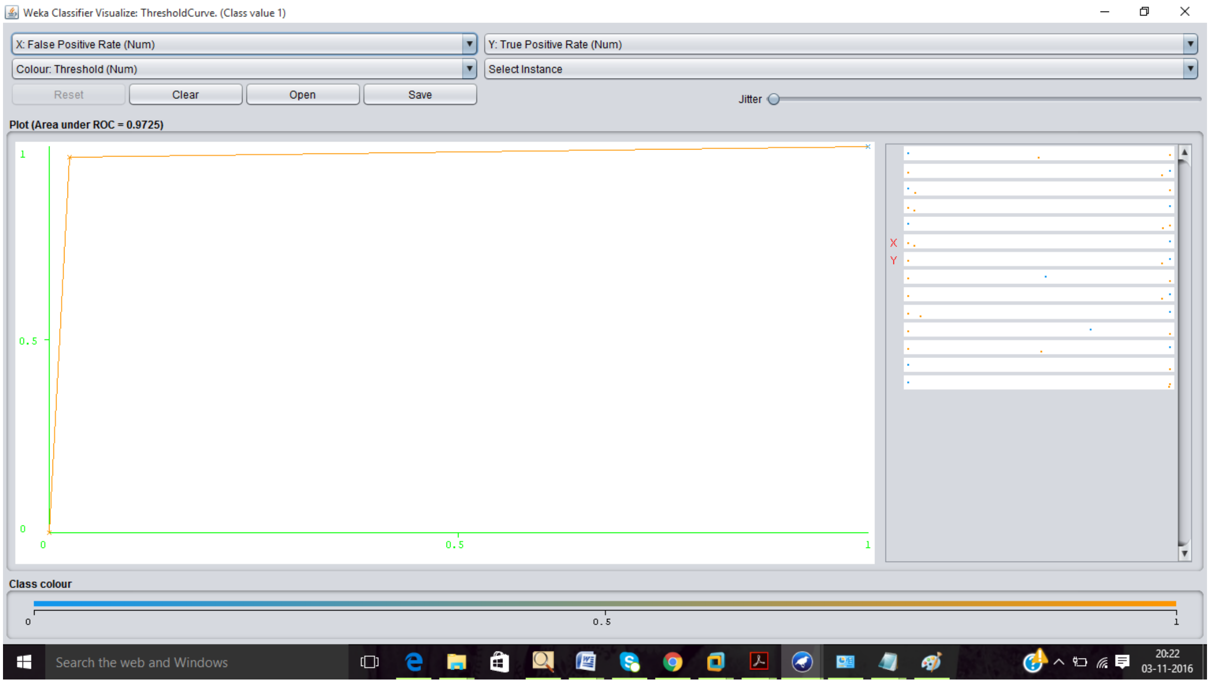
iii) TWO GAUSSIANS DATASET

Screenshot: Classes 1 and 2 samples are visualised in blue and red colour respectively

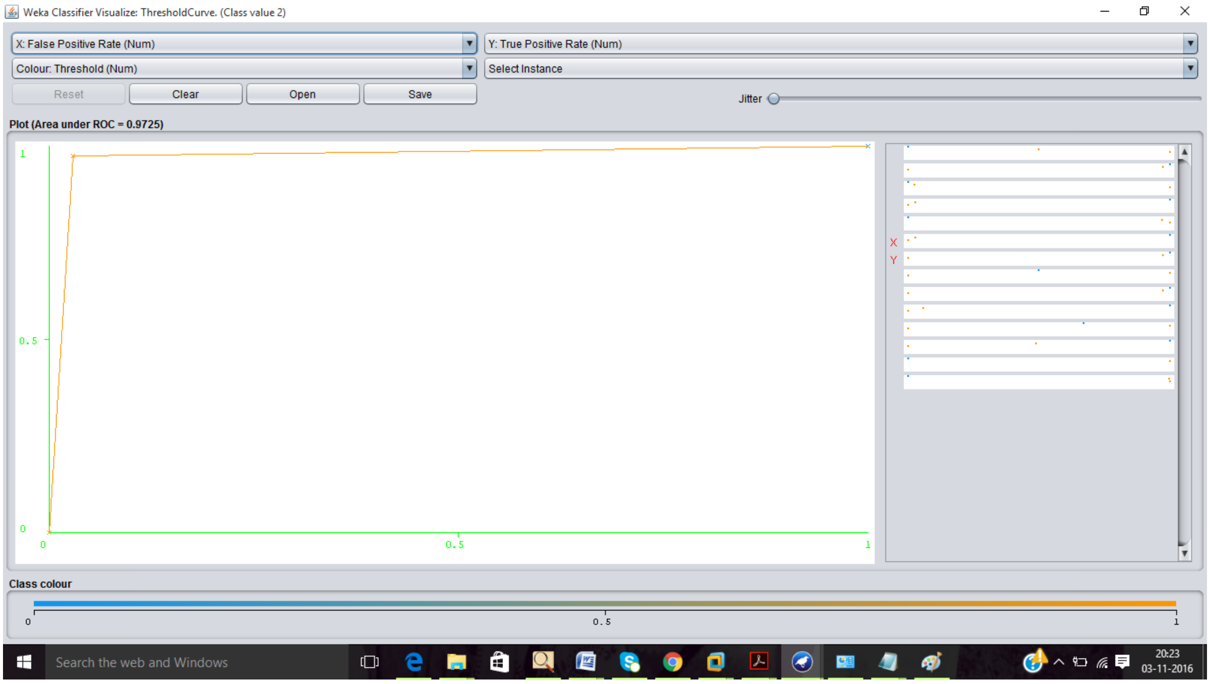


<u>SVM- L(Linear kernel)</u>

Screenshot: For SVM-Linear, Visualizing the threshold curve for Class 1

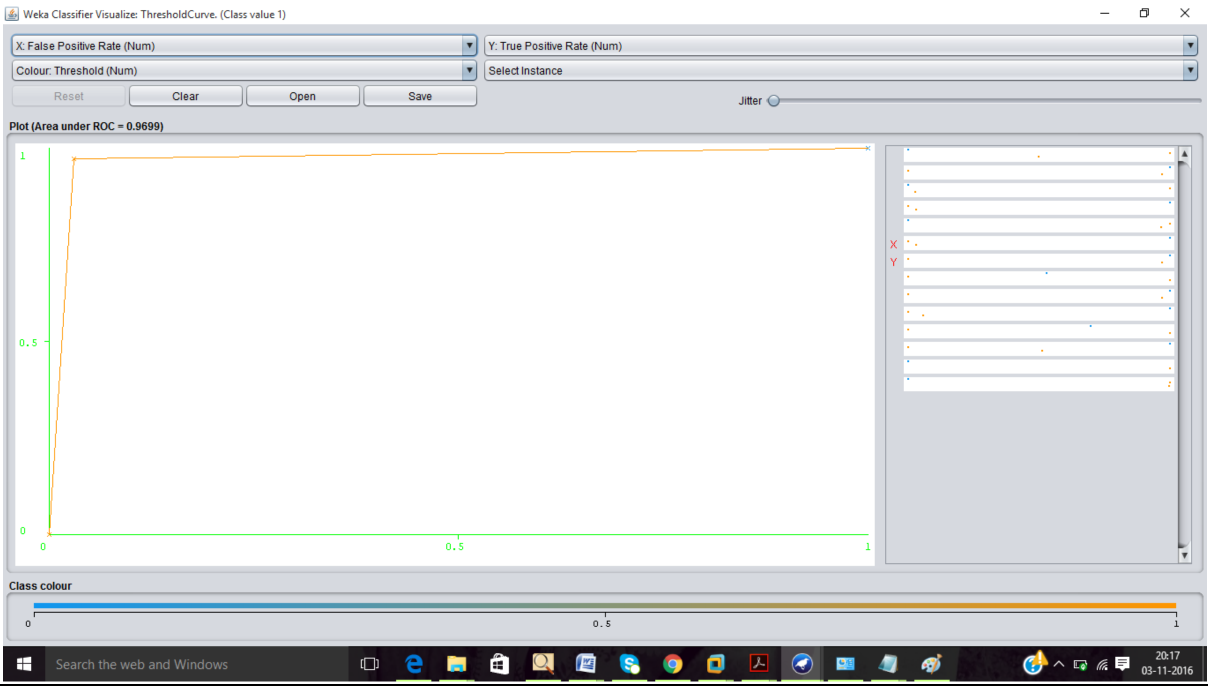


Screenshot: For SVM-Linear, Visualizing the threshold curve for Class 2

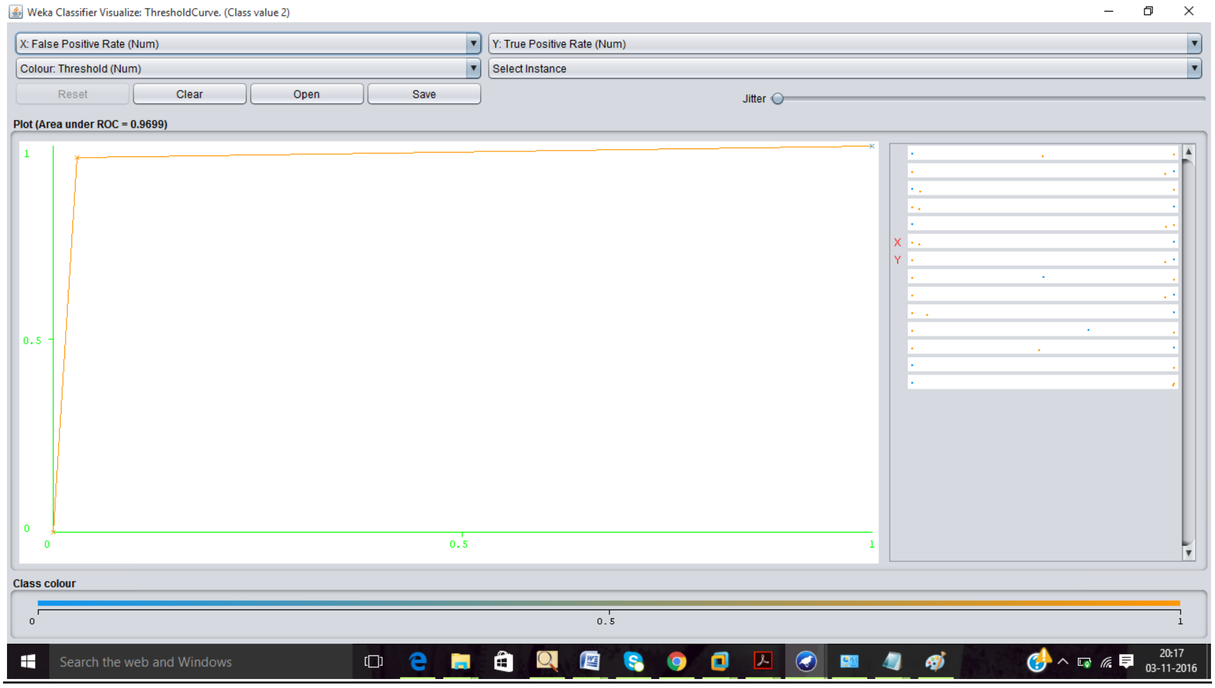


SVM –P (Polynomial kernel)

Screenshot: For SVM-Polynomial, Visualizing the threshold curve for Class 1

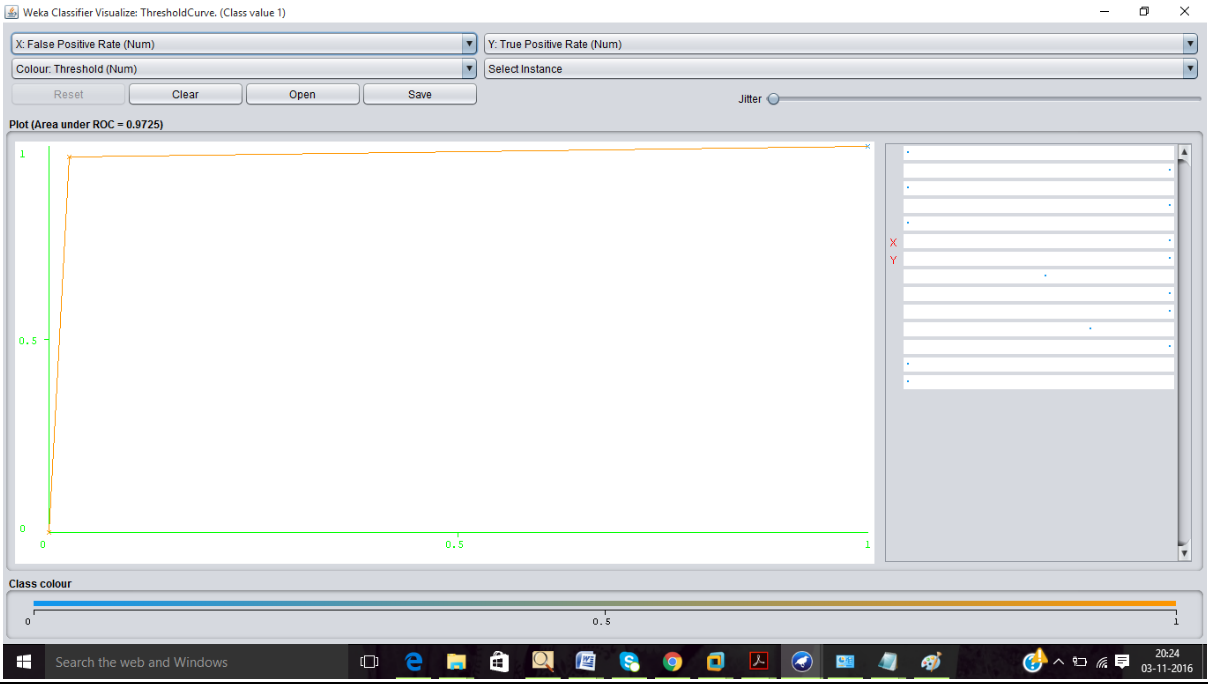


Screenshot: For SVM-Polynomial, Visualizing the threshold curve for Class 2

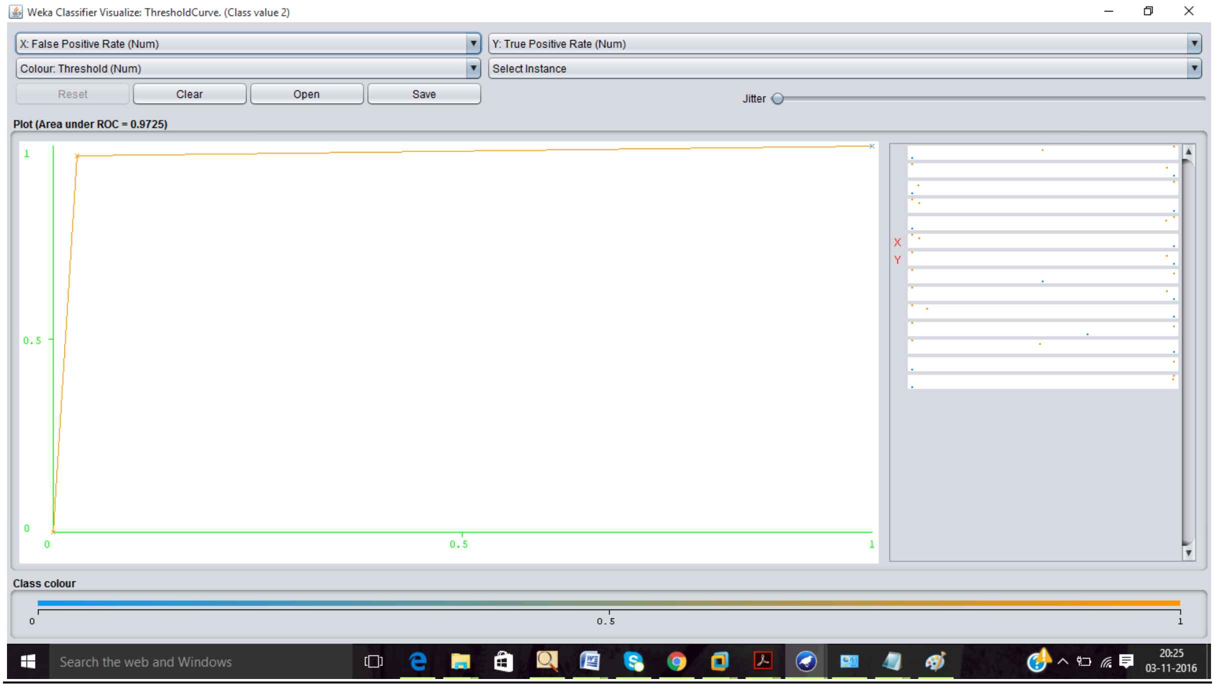


<u>SVM- R (Radial Base Function kernel)</u>

Screenshot: For SVM-RBF, Visualizing the threshold curve for Class 1

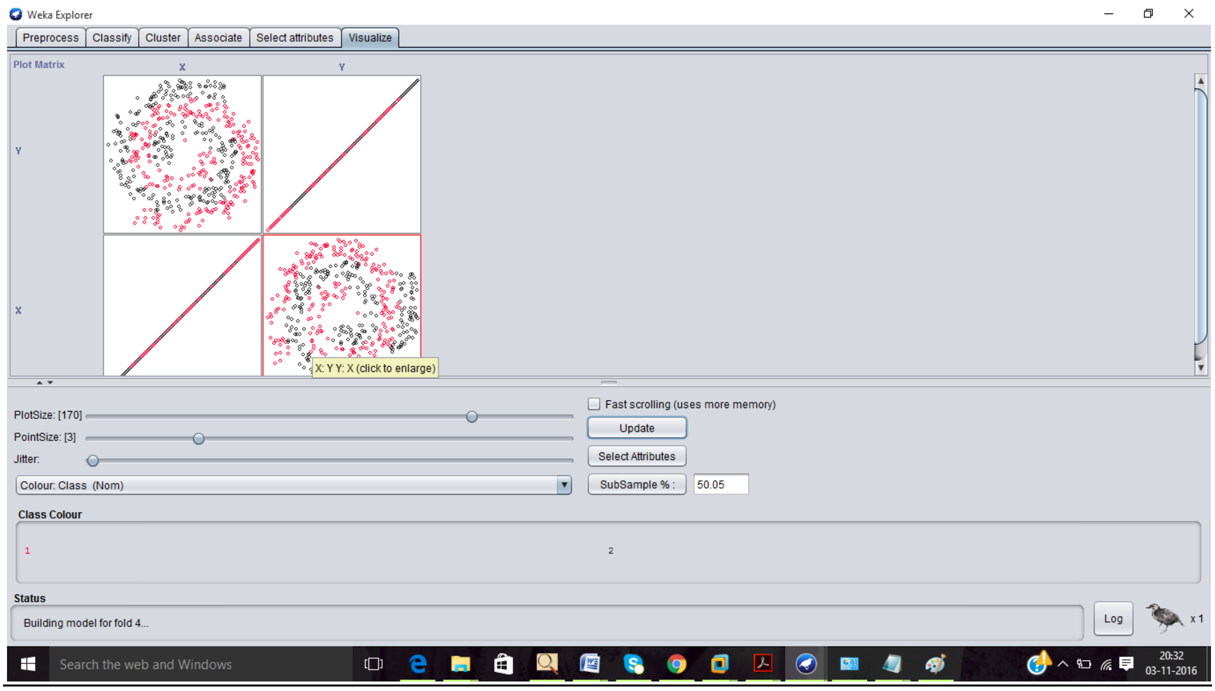


Screenshot: For SVM-RBF, Visualizing the threshold curve for Class 2



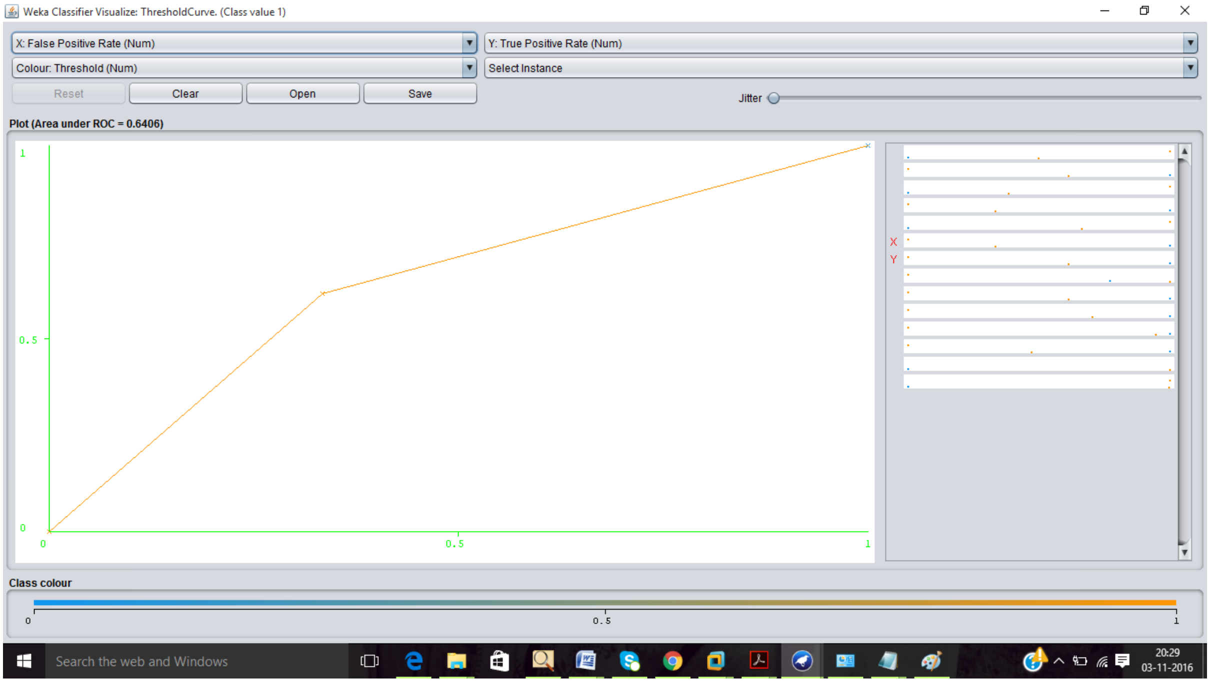
iv) TWO SPIRALS DATASET

Screenshot: Classes 1 and 2 samples are visualised in red and grey colour respectively

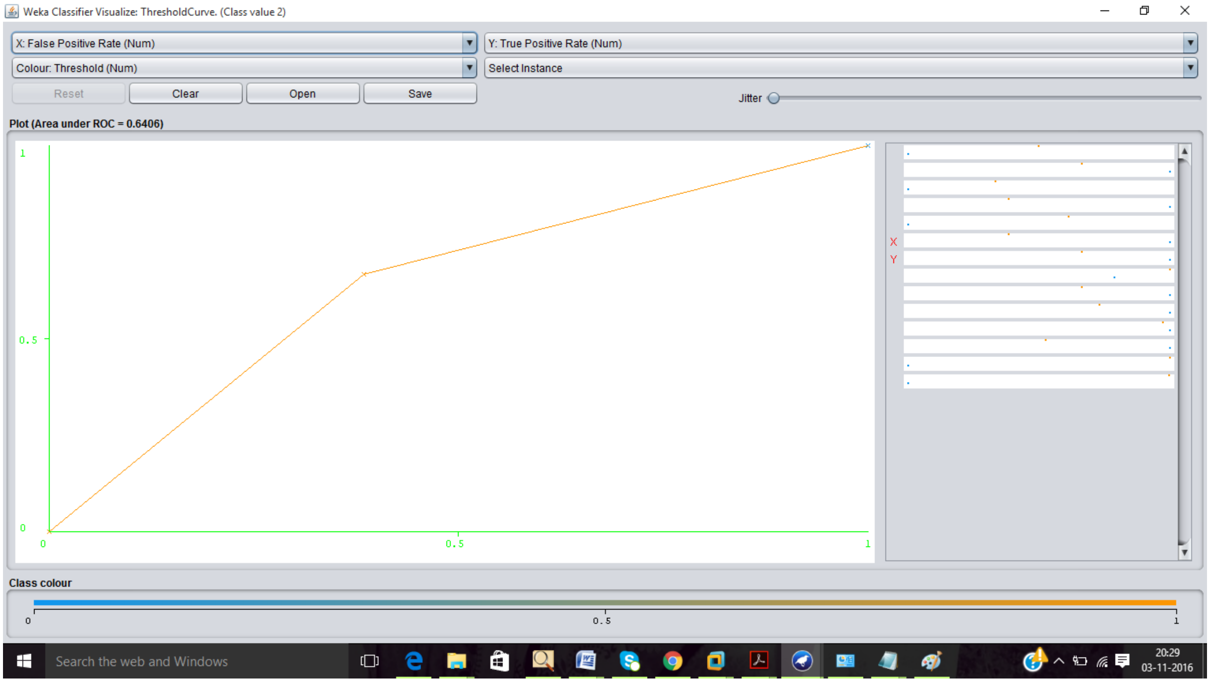


<u>SVM- L(Linear kernel)</u>

Screenshot: For SVM-Linear, Visualizing the threshold curve for Class 1

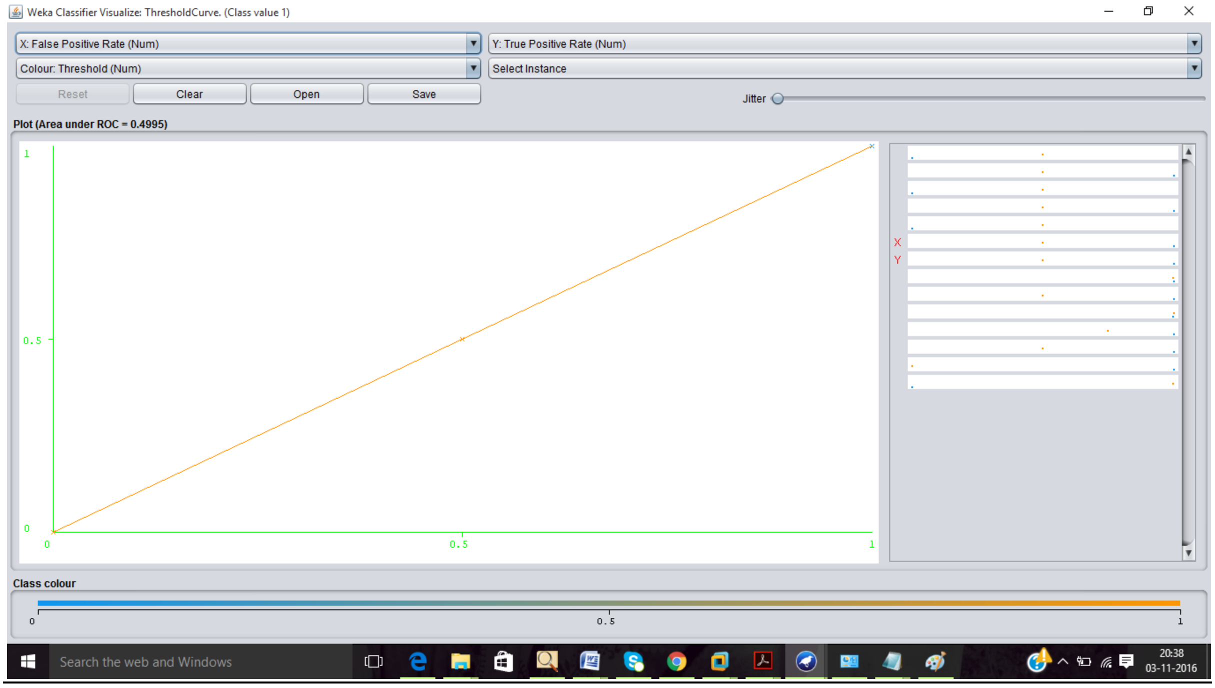


Screenshot: For SVM-Linear, Visualizing the threshold curve for Class 2

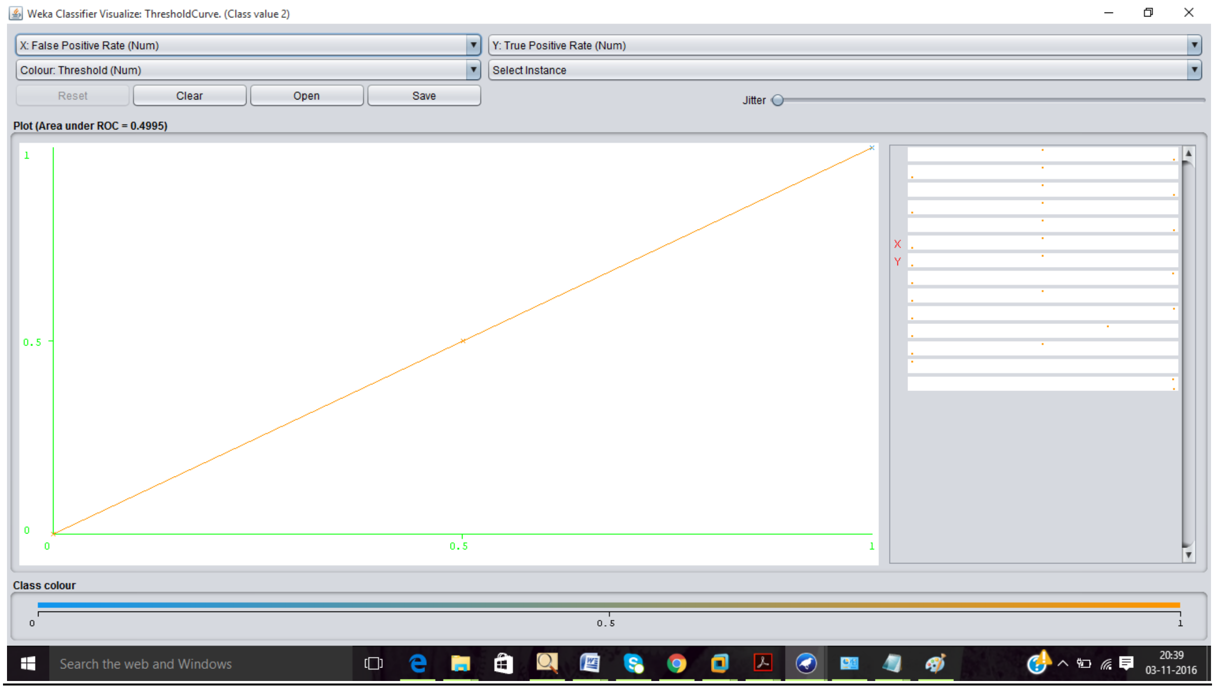


<u>SVM –P (Polynomial kernel)</u>

Screenshot: For SVM-Polynomial, Visualizing the threshold curve for Class 1

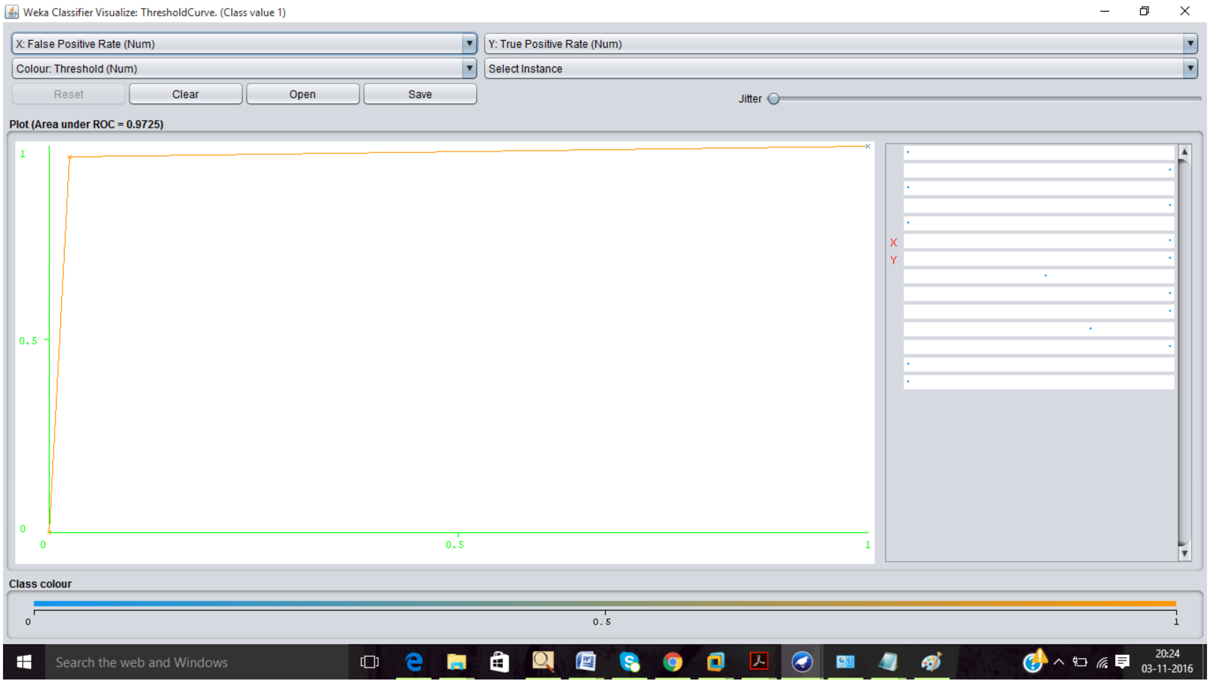


Screenshot: For SVM-Polynomial, Visualizing the threshold curve for Class 2

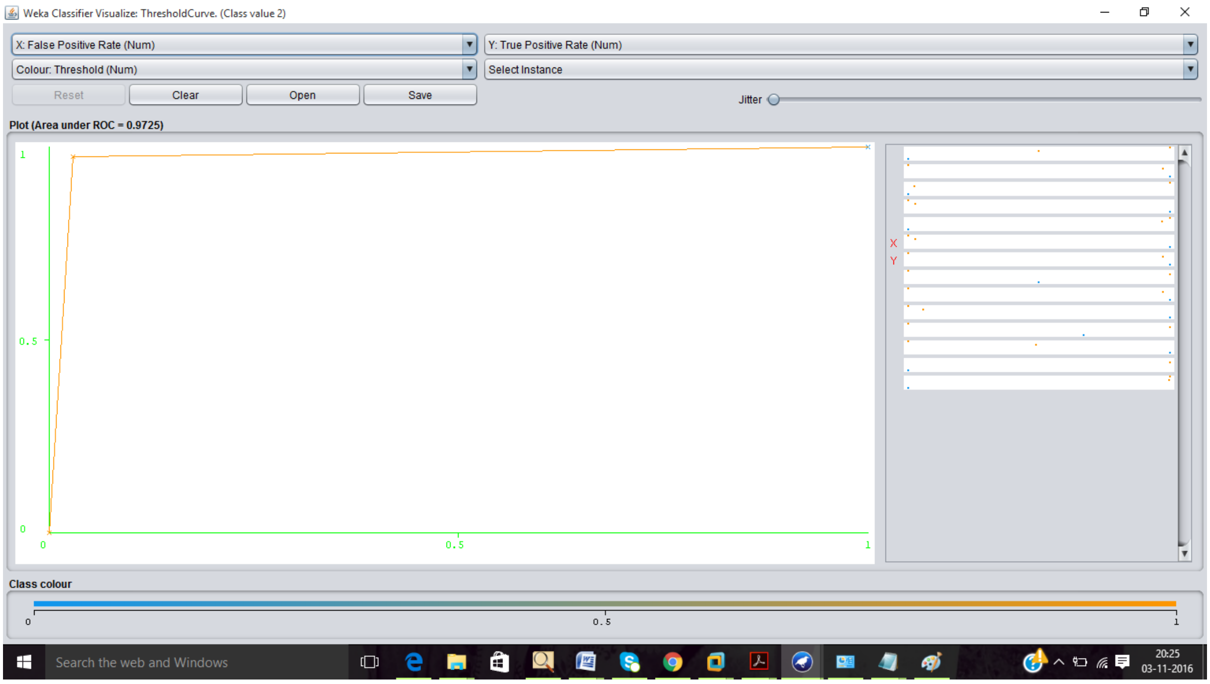


<u>SVM- R (Radial Base Function kernel)</u>

Screenshot: For SVM-RBF, Visualizing the threshold curve for Class 1



Screenshot: For SVM-RBF, Visualizing the threshold curve for Class 2



7) **Comparison between SVM Classifiers and K-NN Classifier**

In terms of Performance

<u>Table: Comparison of SVM and K-NN classifier for various datasets in terms of Accuracy</u>

Classifiers	**SVM**			**K-NN**
Datasets	**SVM-L**	**SVM-P**	**SVM-R**	
Cluster In Cluster	53.4535%	62.2623%	100%	100%
Half Kernel	73.5736%	63.4635%	100%	100%

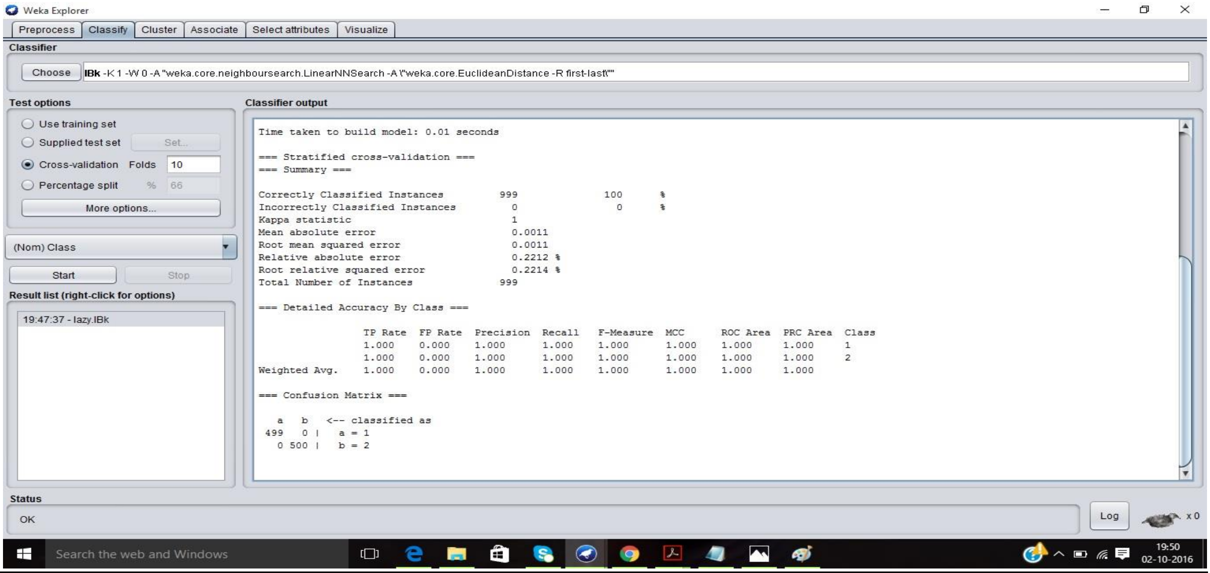
Two Gaussians	97.2431%	96.9925%	97.2431%	95.99%
Two Spirals	64.0641%	99.9499%	95.1952%	94.49%

In terms of AUC

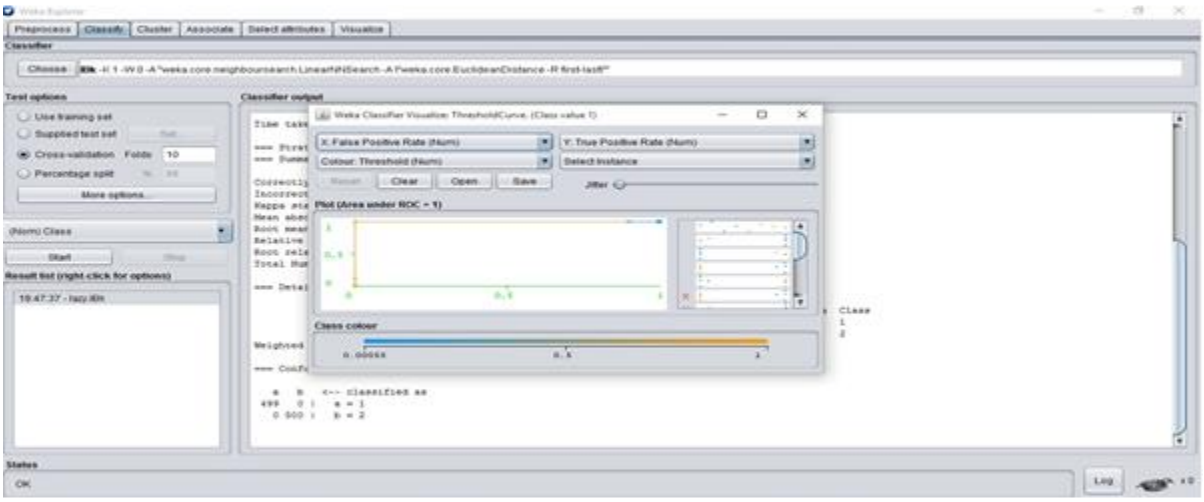
K-NN Classifier

i) Cluster in Cluster Dataset:

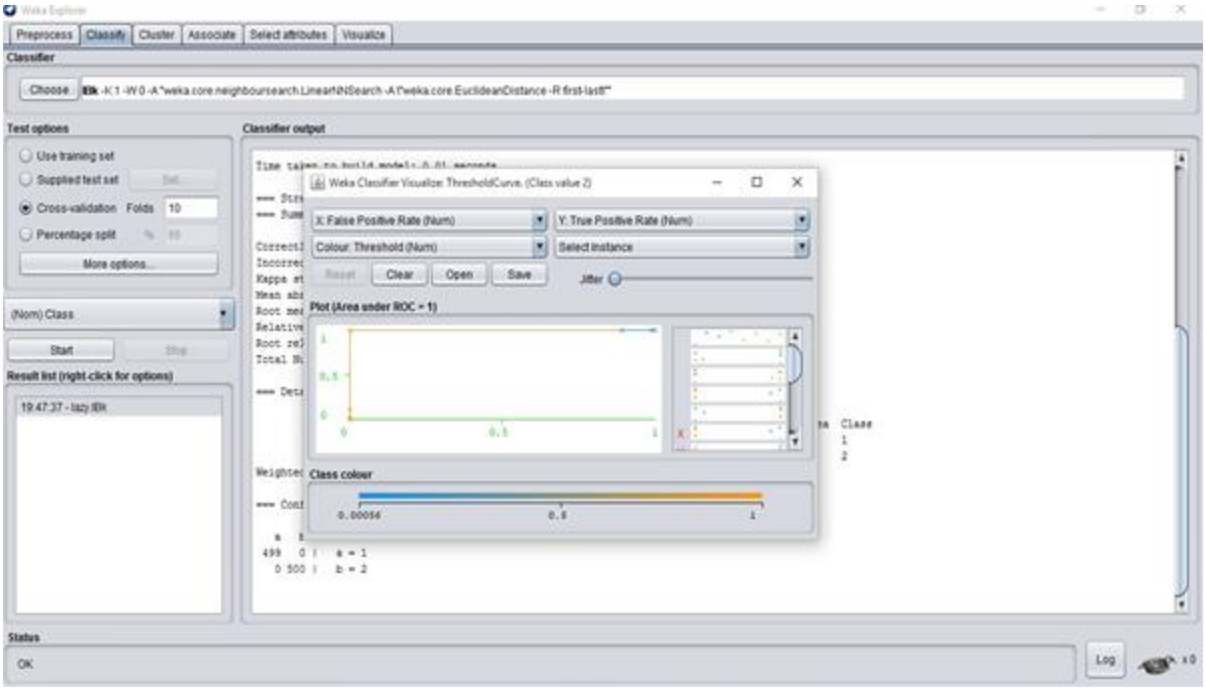
Screenshot: When K=1, Accuracy is 100%



Screenshot: For K=1, Visualizing the threshold curve for Class 1

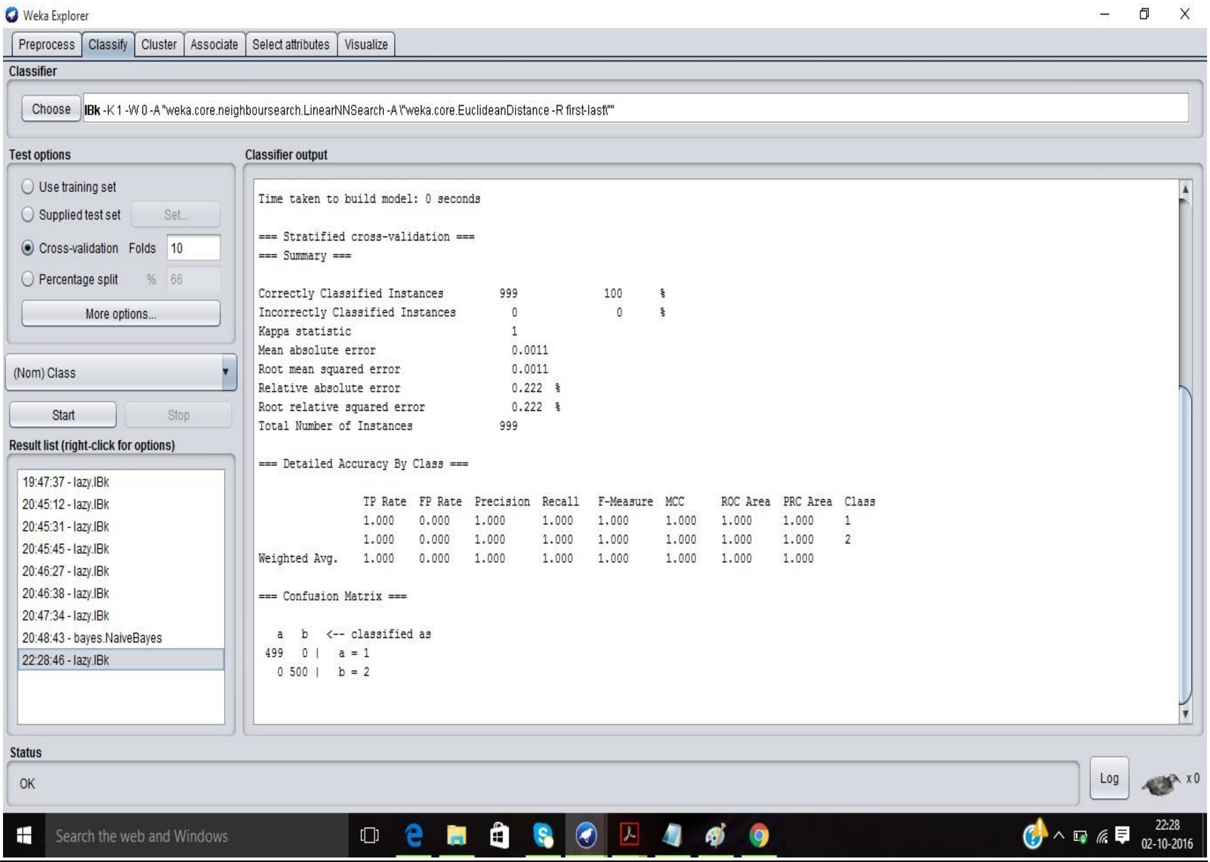


Screenshot: For K=1, Visualizing the threshold curve for Class 2

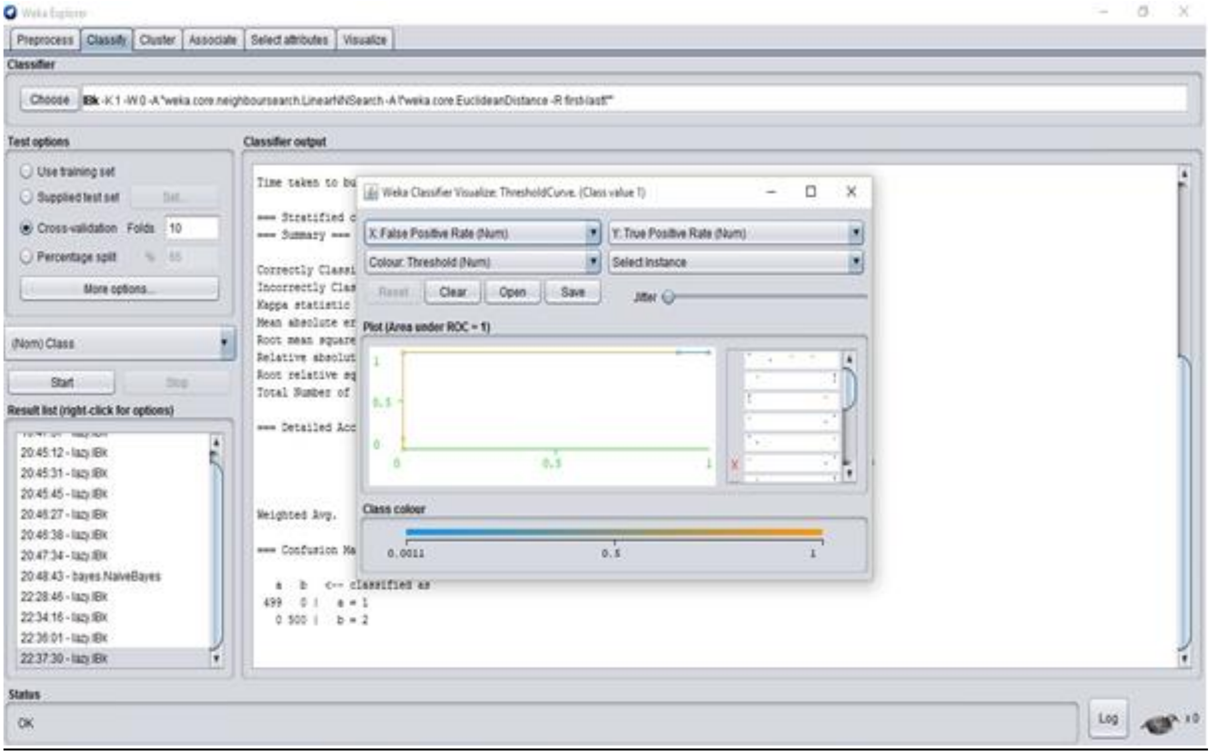


ii) Half Kernel Dataset:

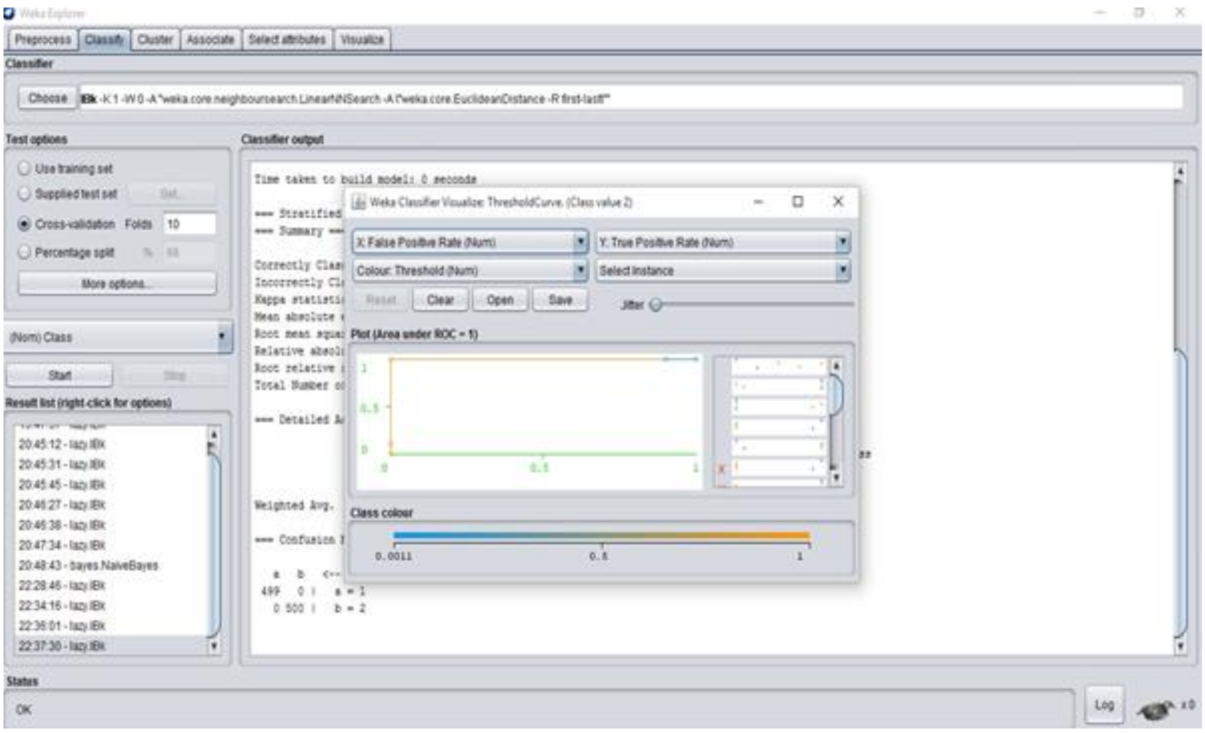
Screenshot: When K=1, Accuracy is 100%



<u>Screenshot: For K=1, Visualizing the threshold curve for Class 1</u>

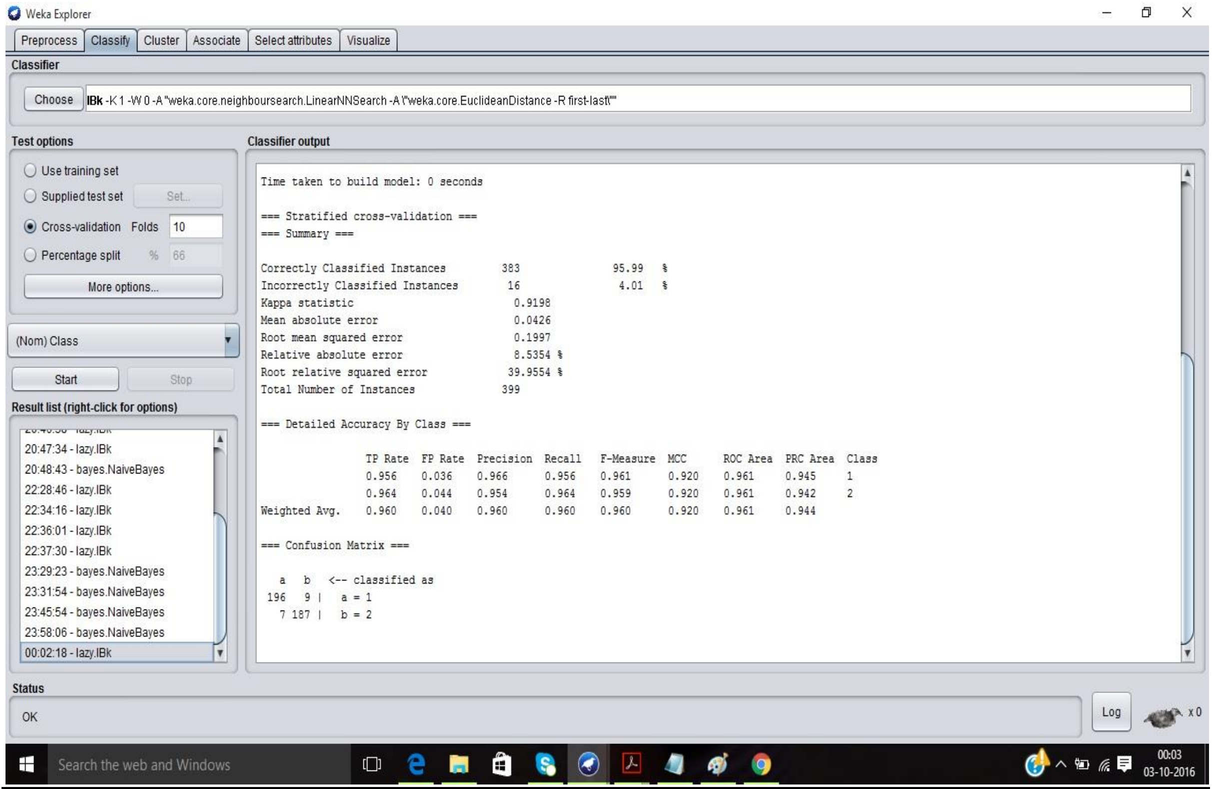


Screenshot: For K=1, Visualizing the threshold curve for Class 2

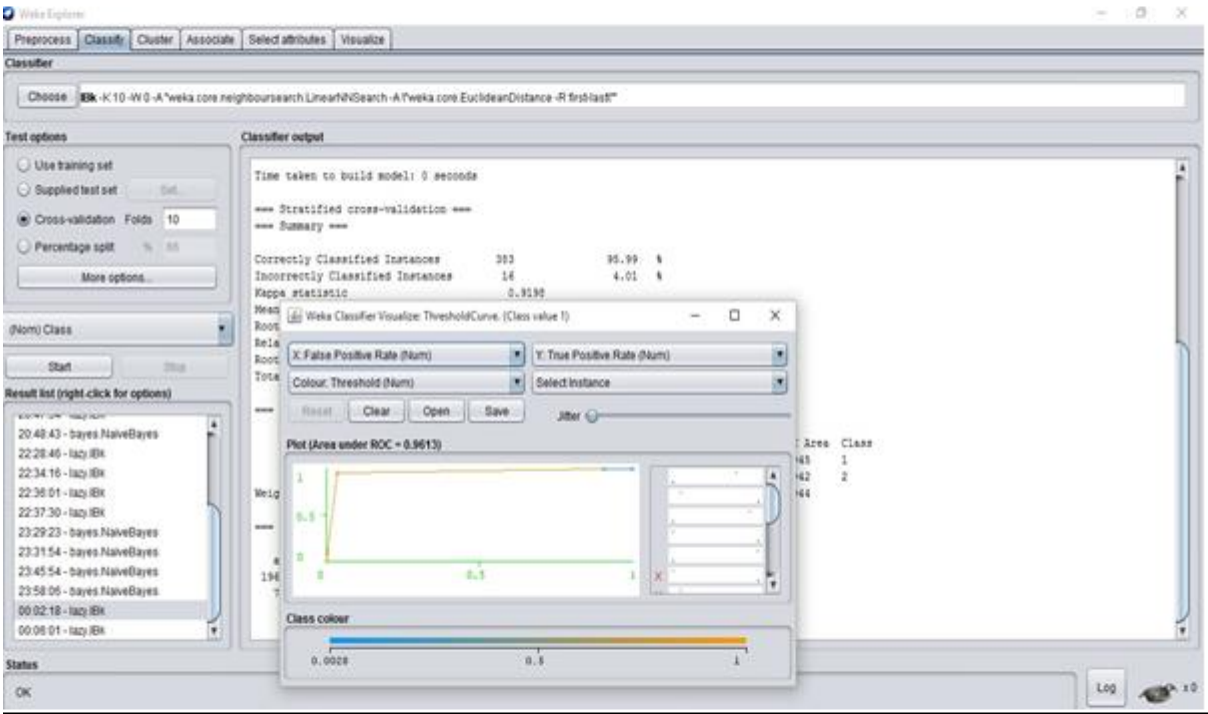


iii) Two Gaussians Dataset:

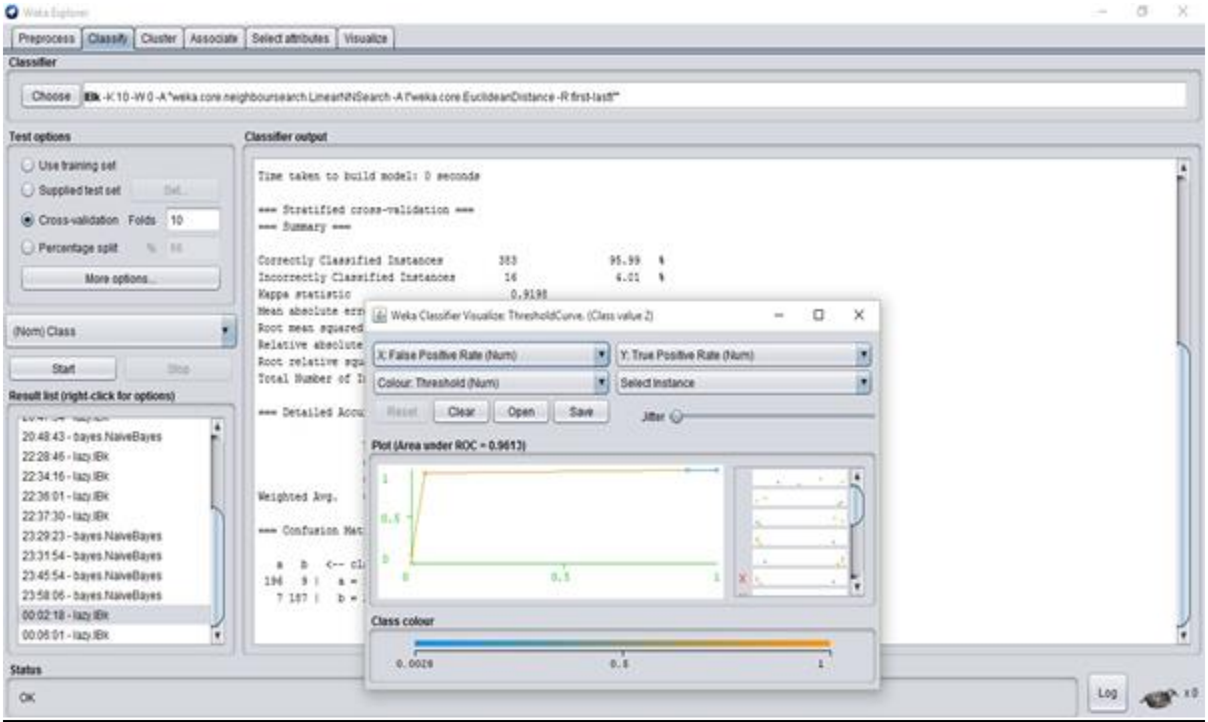
Screenshot: When K=1, Accuracy is 95.99%



Screenshot: For K=1, Visualizing the threshold curve for Class 1

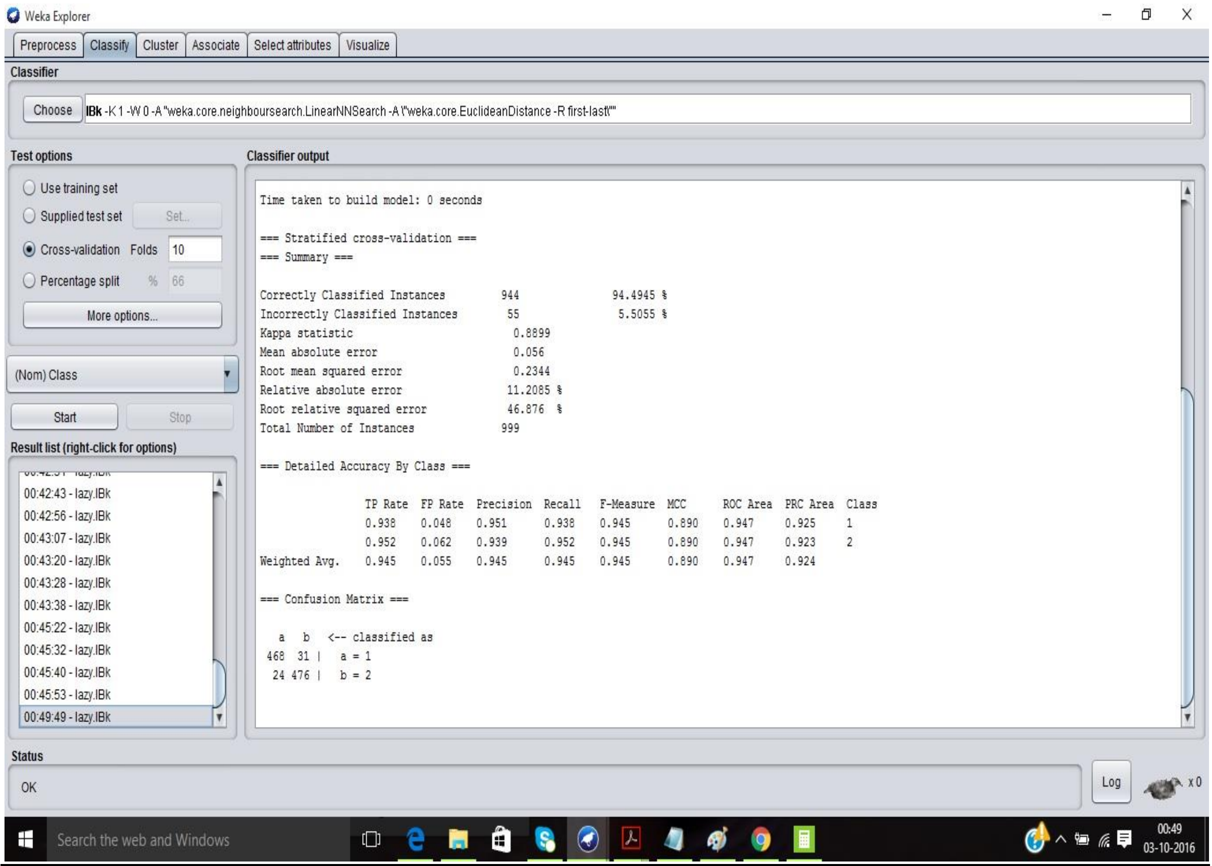


Screenshot: For K=1, Visualizing the threshold curve for Class 2

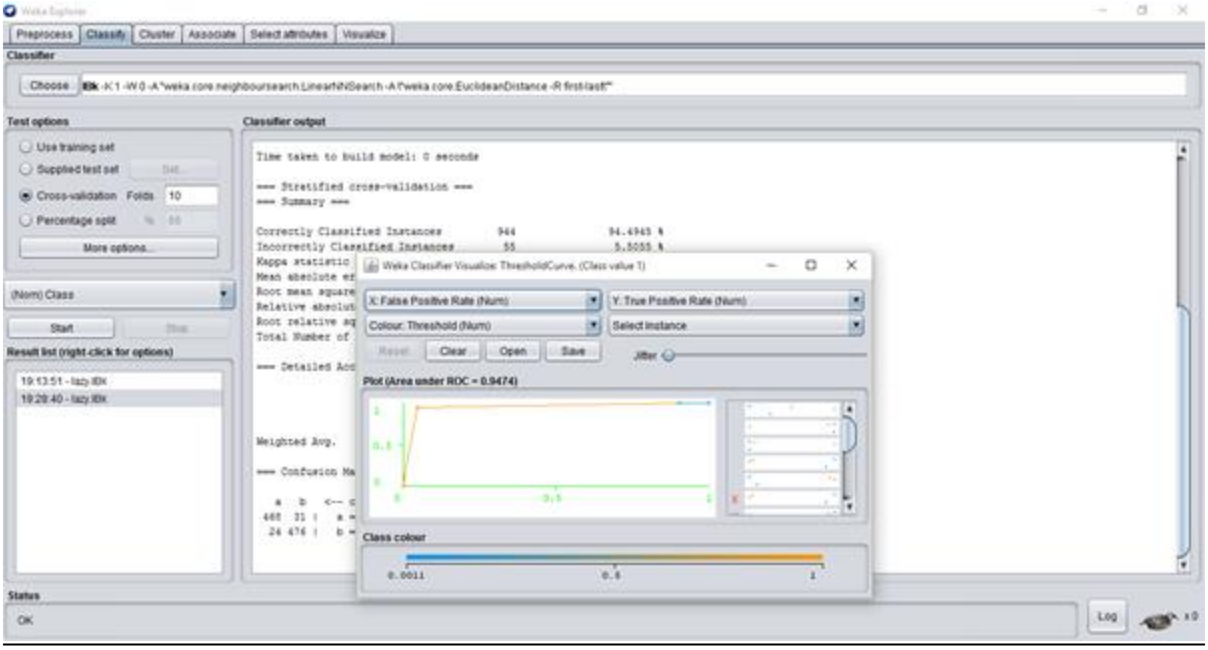


iv) Two Spirals Dataset:

Screenshot: When K=1, Accuracy is 94.49%



Screenshot: For K=1, Visualizing the threshold curve for Class 1



Screenshot: For K=1, Visualizing the threshold curve for Class 2

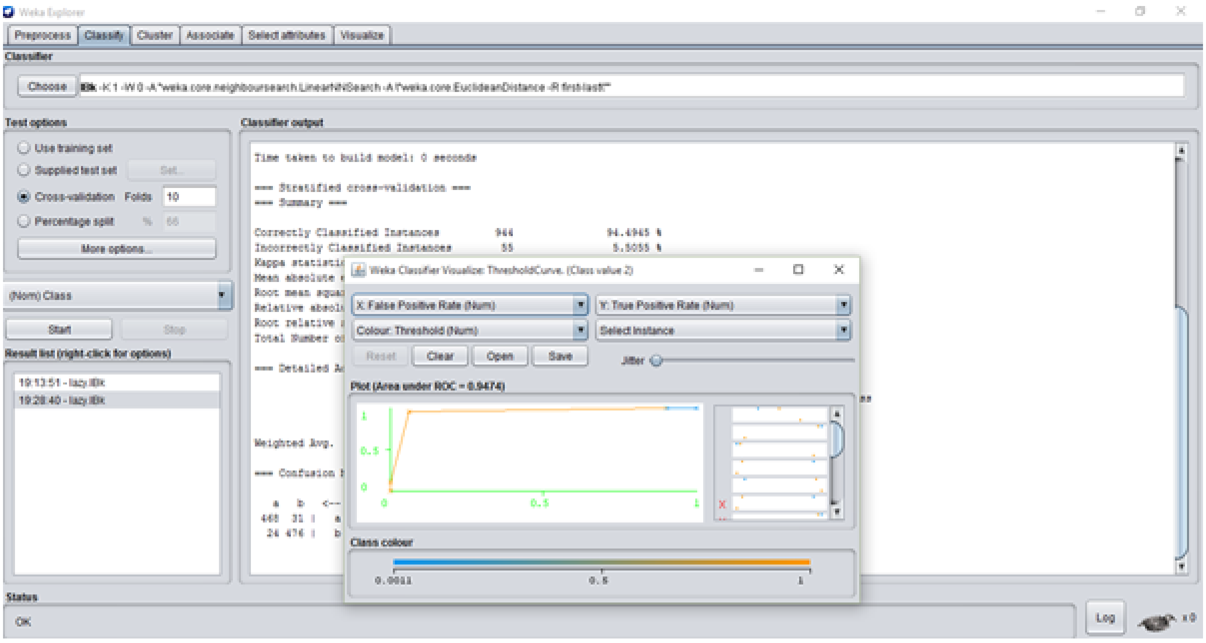


Table: Comparison of SVM and K-NN classifier for various datasets in terms of Accuracy

Classifiers	**SVM**			**K-NN**
Datasets	**SVM-L**	**SVM-P**	**SVM-R**	
Cluster In Cluster	0.5347	0.623	1	1
Half Kernel	0.7359	0.6348	1	1
Two Gaussians	0.9725	0.9699	0.9725	0.9613
Two Spirals	0.6406	0.4995	0.9725	0.9474

SVM –RBF kernel performs better in classification while comparing it with K-NN classifier.

SVM Classifier is better than K-NN:

- SVM- Radial Basis Function Kernel produces better classification, performance metrics and AUC than K-NN based on the above results. The choice of kernel depends on the problem. But, SVM is very popular and widely used algorithm.
- SVM is suitable for small training datasets and large number of features. Support vectors have excellent generalization power. It intends to find the samples, which are difficult to classify. However, for large number of datasets, K-NN classifier will serve better.
- K-NN is non-linear by nature and it can detect both linear and non-linear even when it has to classify with many data points. In lower dimensional space, if there are many data points then K-NN is the ideal choice and if there are few data points in high dimensional space then SVM is ideal.
- K has to be tuned as it determines the distance between the samples. In SVM, Cost C has to be tuned in order to optimize the parameters.
- The error of SVM depends on the number of support vectors only (n_s) whereas K must be odd in K-NN classifier in order to avoid ties and misclassification.
- SVM RBF kernel is much better in unpredictable situations and it relies on the gamma value while tuning for optimization.