Development of Automated Methods for Using Satellite Imagery to Monitor Changes in Vegetative Communities

Report for Quarter 3/Task 4 in 2021

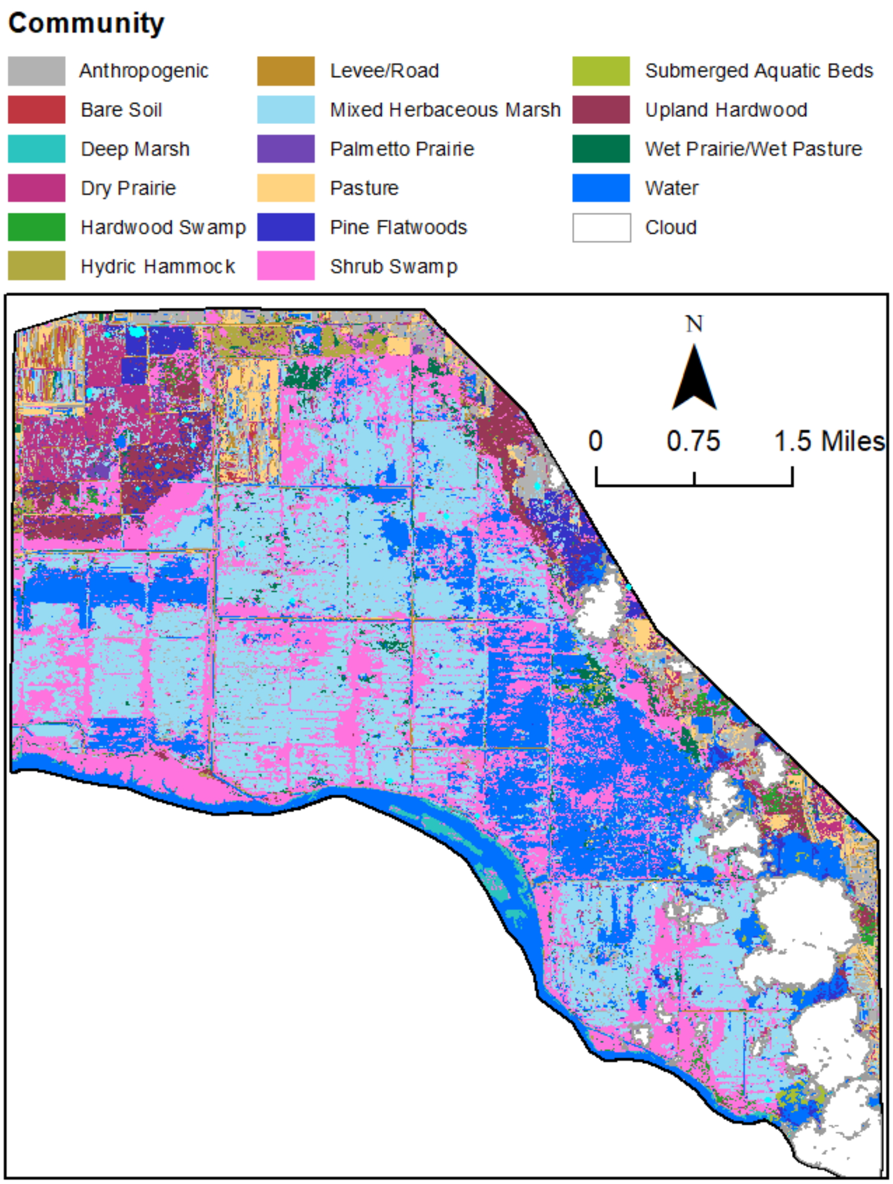
Submitted to: St. Johns River Water Management District

By

Caiyun Zhang, PhD

Department of Geosciences, Florida Atlantic University

777 Glades Road, Boca Raton, FL 33431, USA, Tel: 561-297-2648; Email: czhang3@fau.edu



Classified Vegetation Map for East Lake Apopka North Shore (LANS) Area from Machine Learning Ensemble Analysis

Contents

In this Quarter, we remapped Ocklawaha Prairie Restoration Area (OPRA) and Lake Apopka North Shore Area (LANS) using segmentation results from eCognition software package, and 8-band WorldView-2 (WV-2) imagery. LiDAR DEM was also included in the classification because LiDAR DEM could improve classification accuracy as reported in previous quarters. We also compared 4-band vs. 8-band WV-2 in mapping accuracy for OPRA. In the last Quarter, data deliverables were produced using segmentation results from OTB/QGIS. In this report, we provide the eCognition segmentation results, mapping accuracy, and final classification maps. We used the same training points collected in the last Quarter and identified corresponding training segments. We also developed detailed tutorials for object-based machine learning training, classification, and mapping using ArcGIS, eCognition, and open source WEKA (Appendix 1). We are developing tutorials using R and Python for machine learning training and classification, and ensemble analysis. We will provide the tutorials in next project year.

1. Results for OPRA

1.1 Segmentation result

The segmentation results from eCognition are provided in Figure 1. There were 73,001 segments for OPRA with a greatly reduced data volume compared to the previous OTB/QGIS segmentation which produced 669,720 segments. Spectral and texture features were exported from eCognition into ArcGIS for training data processing and preparation.

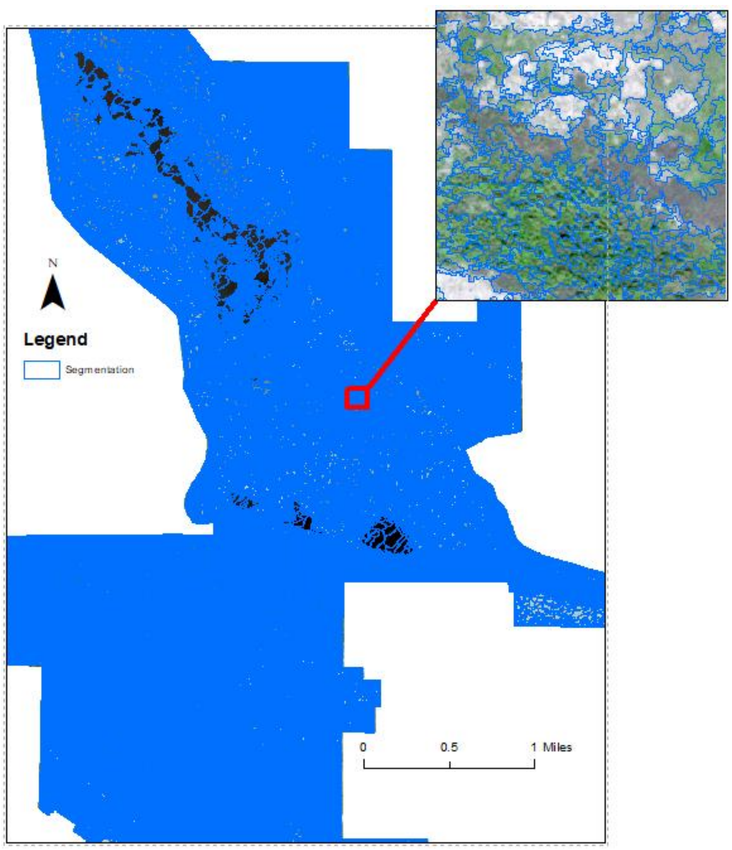


Figure 1 Segmentation from eCognition with a total of 73,001 segments for OPRA.

1.2 Model performance

The performance of tested machine learning classifiers Artificial Neural Network (ANN), Support Vector Machine (SVM), Random Forest (RF), and *k* Nearest Neighbor (kNN) using 8-band/4-band WV-2 imagery is displayed in Table 1. For this area, the accuracy was slightly reduced compared to OTB/QGIS segmentation classification results (84.7%) but produced an overall accuracy (OA) of 80.3% using SVM and ensemble analysis which combined the outputs from ANN, SVM, and RF. The slightly reduced accuracy was mainly caused by the reduced number of training samples for UH (Upland Hardwood) (1004 vs. 4156) used here. OTB/QGIS produced more segments than eCognition, thus it resulted in more training samples for UH which is the major community for this area. Identification of UH had a higher accuracy (>90%). More training samples for this community could amplify the OA in QGIS. The application of 4-band generated a comparable accuracy to the application of 8-band in the classification (Table 1).

Table 1. Performance of each machine learning classifier and ensemble analysis for OPRA (8-band/4-band); EA is ensemble analysis of ANN, SVM, and RF.

	ANN	kNN	SVM	RF	EA
Overall Accuracy (%)	77.5/77.0	73.5/73.1	80.3/78.0	79.7/79.7	80.3
Kappa Value	0.74/0.73	0.69/0.69	0.77/0.74	0.76/0.77	0.77

1.3 Classified maps

Classification maps from machine learning algorithms, and ensemble analysis are displayed in Figure 2. In general, all classifiers produced a consistent spatial pattern with the existing map, but classification maps provide a more realistic/natural edge of each patch and presence of small patches within large communities. Pine Plantation was delineated from the original Upland and mainly distributed in the lower right corner. For most regions, three classifiers (ANN, SVM and RF) generated consistent classification, as shown in green in the uncertainty map in Figure 2 (g). Red regions show a "warning sign" where classification from three classifiers were completely inconsistent.

1.4 Analysis

For this area, an interesting finding is that in ANN classification of OTB/QGIS segments, water was misclassified as Pasture, but here using eCognition, Open Water (OW) was accurately identified in ANN (Figure 2e) and consistent with other classifiers. Similar to the OTB/QGIS results, the error matrix, showing the performance of each classifier in identifying individual classes, demonstrates that Bayhead/gall (BG), Cypress (CY), Forested Flatwood Depressions (FD) and Hydric Hammock (HH) had a lower user's accuracy (UA). BG and HH were confused with Upland Hardwood (UH), while Palmetto Prairie (PP) was confused with Wet prairie/wet pasture (WP). Much of the confusion was due to the small nature of most of these communities, resulting in only a small number of training samples for each community. The error matrix is large and uploaded as a separate Excel file.

Based on the comparison of classification accuracies using 4-band vs. 8 band WV-2 imagery, we found that 4-band achieved a comparable accuracy as 8-band. This is consistent with the results reported in the last Quarter. In addition, the time cost for segmenting 8-band imagery and exporting the attributes in eCognition was doubled. The data volume was also doubled. Since this result mirrored earlier results with other imagery, further testing using LANS imagery was not deemed necessary. We recommend using 4-band WV-2 imagery in future mapping endeavors.

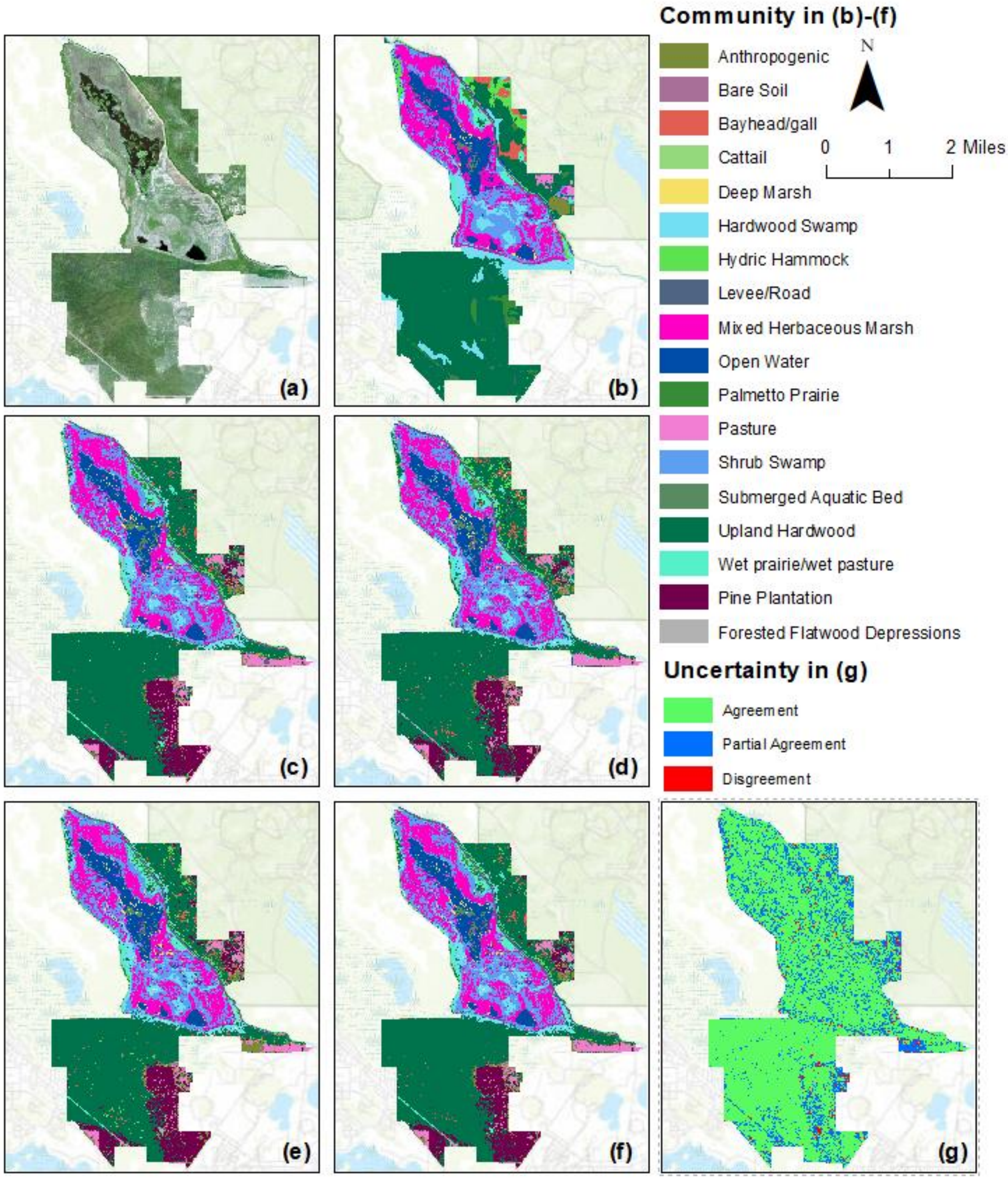


Figure 2 (a) WV-2 pansharpened imagery; (b) reference map from aerial photography; Object-based classifications from (c) SVM, (d) RF, (e) ANN, and (f) Ensemble analysis. (g) Uncertainty from ANN, SVM, and RF for OPRA.

2. Results for West LANS

2.1 Segmentation result

The segmentation results for west LANS from eCognition are provided in Figure 3. There were 65,650 segments for western LANS and the data volume was largely reduced compared to the segmentation from OTB/QGIS which produced a total of 141,154 segments. Spectral and texture features were exported from eCognition into ArcGIS for training data processing and preparation.

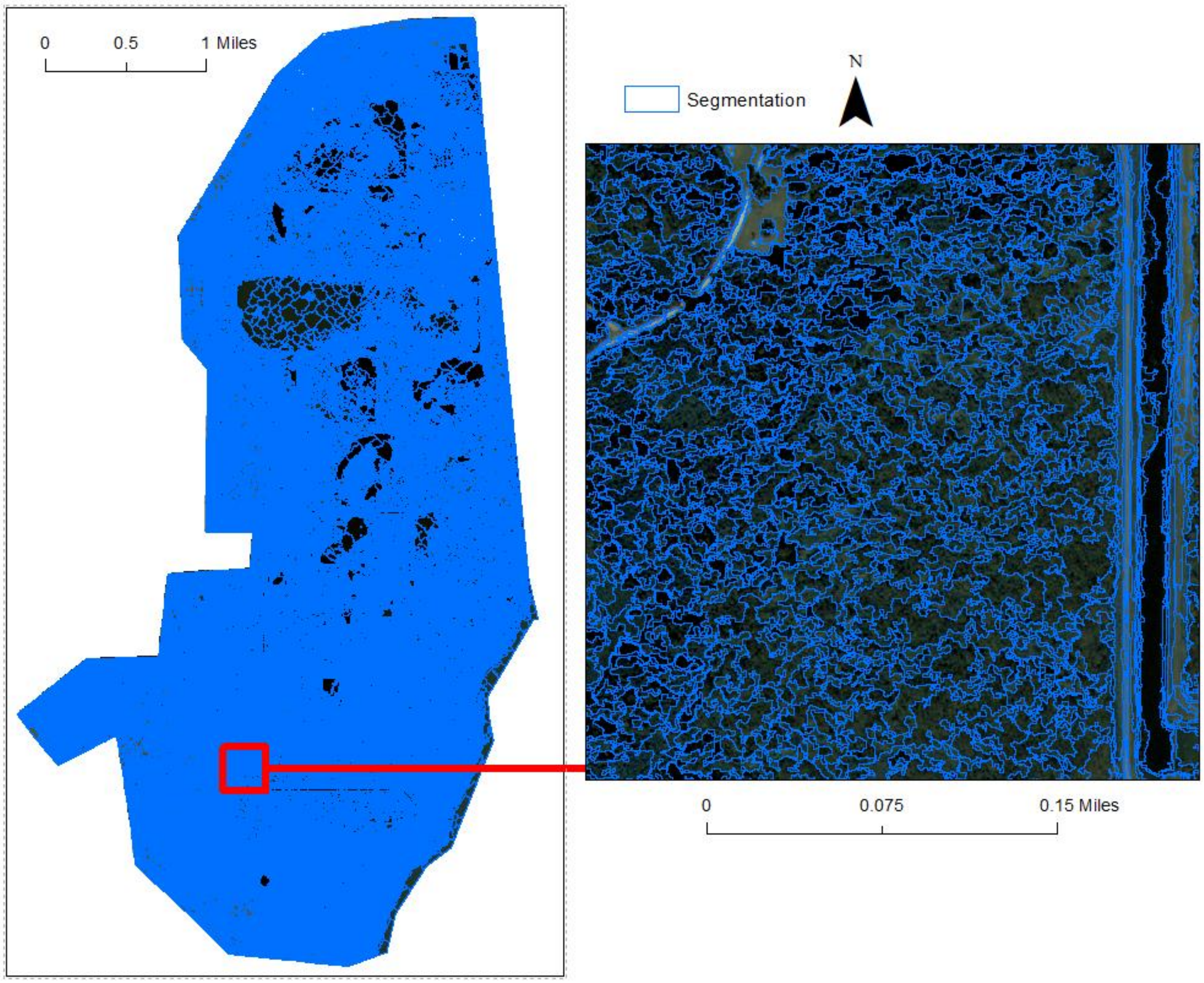


Figure 3 Segmentation from eCognition with a total of 65,650 segments for west LANS.

2.2 Model performance

The performance of machine learning classifiers ANN, SVM, RF, and kNN is displayed in Table 2. For the west LANS, the accuracy was greatly improved using the eCognition segmentation result (~10%) compared to the application of OTB/QGIS segmentation for classifications from SVM (76.1% vs. 66.4%) and ANN (75.7% vs. 65.3%), but for RF, the accuracy was only slightly increased from 74.3% to 75.7%. The best performance was from the SVM classification which achieved an OA of 76.1% and Kappa value of 0.73. Ensemble analysis (EA) of SVM, ANN, and RF slightly improved the OA to 77.5% and Kappa value to 0.74.

Table 2. Performance of each machine learning classifier and ensemble analysis for west LANS using 8-band imagery.

	ANN	kNN	SVM	RF	EA
Overall Accuracy (%)	75.7	67.9	76.1	75.7	77.5
Kappa Value	0.73	0.64	0.73	0.72	0.74

2.3 Classified maps

Classification maps from machine learning algorithms are displayed in Figure 4. In general, all classifiers produced a consistent spatial pattern with the existing map, but classification maps provide a more realistic/natural edge of each patch and presence of small patches within large communities. Regions covered by cloud could not be mapped. The lower left corner in the original vegetation map was Upland which was further detailed into Bare Soil (BS), Crops (AN), Pine Flatwoods (PF), and Upland Hardwood (UH) in the classification map. The spatial patterns of the classifications were also consistent with the OTB/QGIS results shown in the Quarter 2 report. For most regions, three classifiers generated consistent classification, as shown in green in the uncertainty map in Figure 4 (g). Red regions show a “warning sign” where classification from three classifiers are completely inconsistent.

2.4 Analysis

We analyzed the error matrix to identify communities with confusing spectral features. Classification of Hydric Hammock (HH) and Palmetto prairie (PP) had a very low accuracy. HH was confused with Shrub Swamp (SS), Hardwood Swamp (HS), and Upland Hardwood (UH). PP was confused with HS, and UH. Both HH (covers about 1%) and PP (covers about 0.27%) are minor communities for this area. Limited training samples were available for minor communities. This impacted the effective identification of these minor communities. These findings are consistent with the OTB/QGIS results.

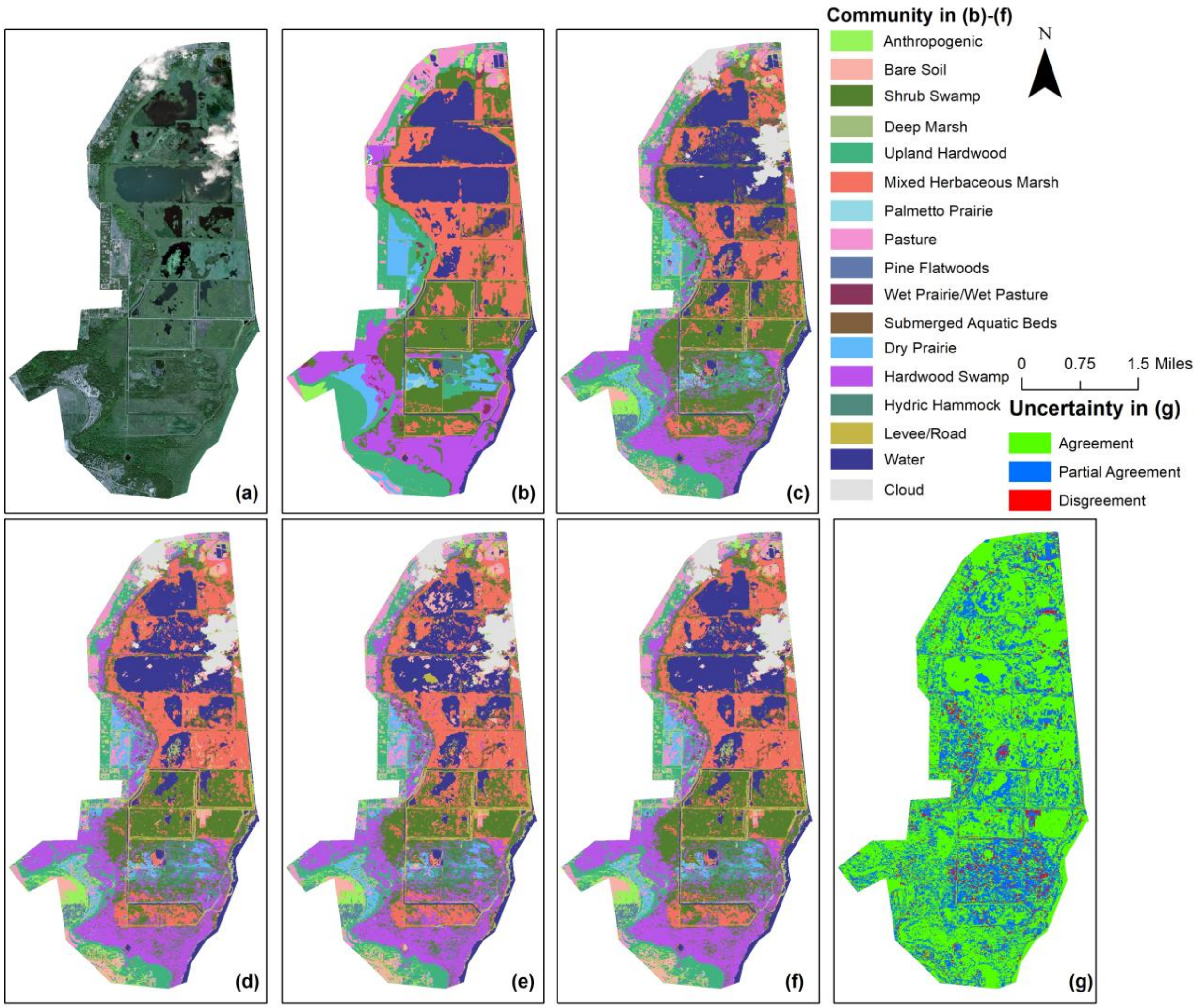


Figure 4 (a) WV-2 pansharpened imagery; (b) reference map from aerial photography; Object-based classifications from (c) SVM, (d) RF, (e) ANN, and (f) Ensemble analysis. (g) Uncertainty from ANN, SVM, and RF for west LANS.

3. Results for East LANS

3.1 Segmentation result

The segmentation results for east LANS from eCognition are provided in Figure 5. There were 145,374 segments for west LANS. Similarly, the data volume was largely reduced compared to the segmentation from OTB/QGIS, which produced a total of 336,234 segments. Spectral and texture features were exported from eCognition into ArcGIS for training data processing and preparation.

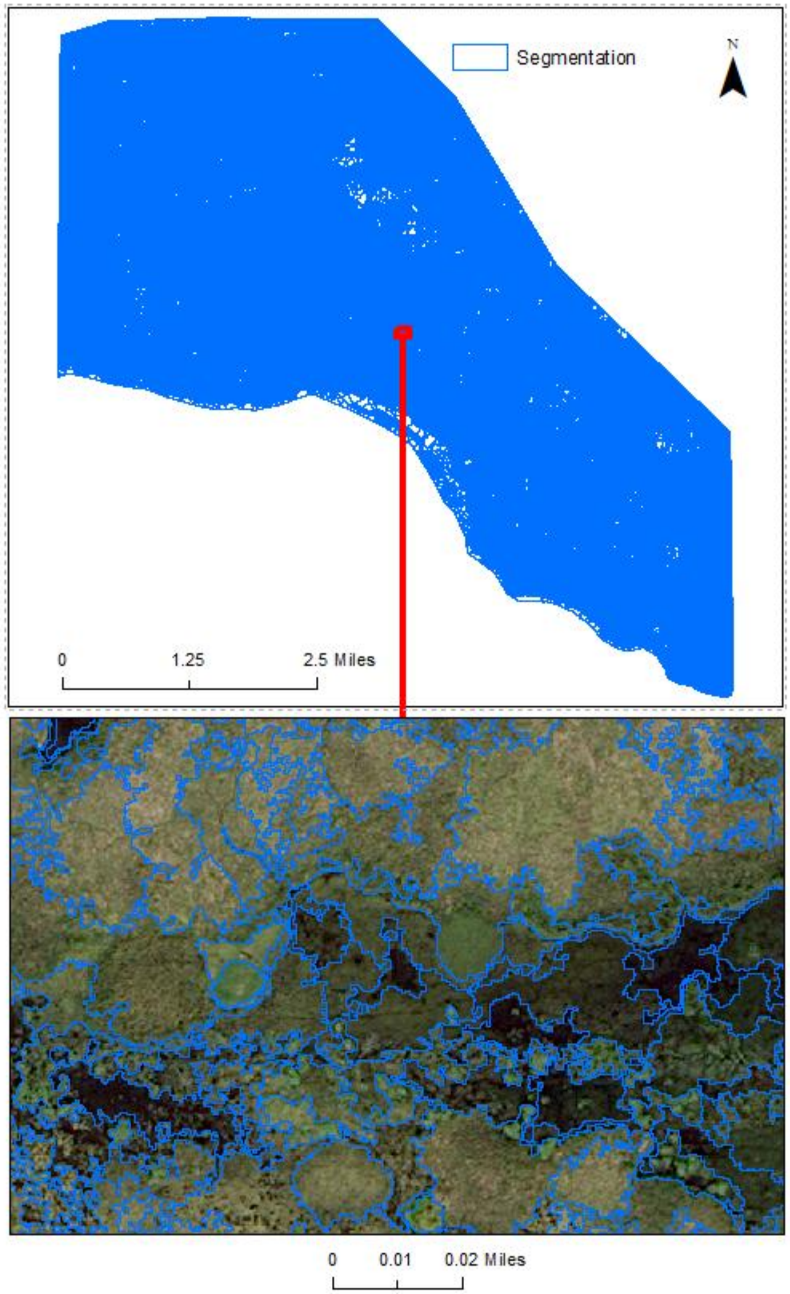


Figure 5 Segmentation from eCognition with a total of 145,374 segments for east LANS.

3.2 Model performance

The performance of classifiers ANN, kNN, SVM, and RF based on 10-fold cross validation is displayed in Table 3. All achieved an OA above 75% except kNN. ANN achieved the best result with an OA of 78% and Kappa value of 0.76. Ensemble analysis of ANN, SVM and RF increased the OA to 79.3%. Similarly, the accuracy was increased around 10% using the eCognition segmentation results compared to the OTB/QGIS segmentation. The error matrix is large due to the classification of 17 communities in this area and is submitted separately as an Excel file.

Table 3. Performance of each machine learning classifier and ensemble analysis for east LANS using 8-band imagery.

	ANN	kNN	SVM	RF	EA
Overall Accuracy (%)	78	68	77	76	80
Kappa Value	0.76	0.64	0.74	0.72	0.77

3.3 Classified maps

Classification maps from machine learning algorithms are displayed in Figure 6. In general, all classifiers produced a consistent spatial pattern with the existing map, but classification maps provide a more realistic/natural edge of each patch and presence of small patches within large communities. Regions covered by cloud could not be mapped. Uplands were further detailed into citrus groves (AN), Pine Flatwoods (PF), and Upland Hardwood (UH) in the classification map. For most regions, three classifiers generated consistent classification, as shown in green in the uncertainty map in Figure 6 (g). Red regions show a "warning sign" where classification from three classifiers are completely inconsistent.

3.4 Analysis

Analysis of the error matrix showed that Hardwood Swamp (HS) was confused with Shrub Swamp (SS) and Upland Hardwood (UH); Palmetto Prairie (PP, a minor community) was confused with Dry Prairie (DP), WP, HM. Submerged Aquatic Beds (SAB, a minor community) was confused with Open Water (OW) which includes water and free-floating plants. These findings are consistent with the results from OTB/QGIS.

For both west and east LANS, applying eCognition segmentation results increased accuracy around 10% compared to the application of OTB/QGIS Large Scale Mean Shift segmentation algorithm. The total number of segments from eCognition was also largely reduced, which made data processing and analysis in later steps easier. Hardwood Swamp, Shrub Swamp, and Upland Hardwood are difficult to differentiate each other. Marsh and prairie are both herbaceous and difficult to differentiate. An inclusion of the canopy height model may help differentiate these communities.

So far all validations were based on the reference samples collected from the original vegetation datasets. District also provided some field collected data for a few areas. In next Quarter, we will validate the classified maps using these field data, map areas with newly collected imagery. We will examine the newly collected imagery and get rough ideas of the change occurred for each area. If large changes were occurred between 2017 and 2021 and training samples are difficult to identify from the 2021 imagery, we will go to the field and collect training samples for some specific areas. In next project year, we will generate vegetation maps from the 2021 imagery and collect some field data to further calibrate the models and validate the final maps.

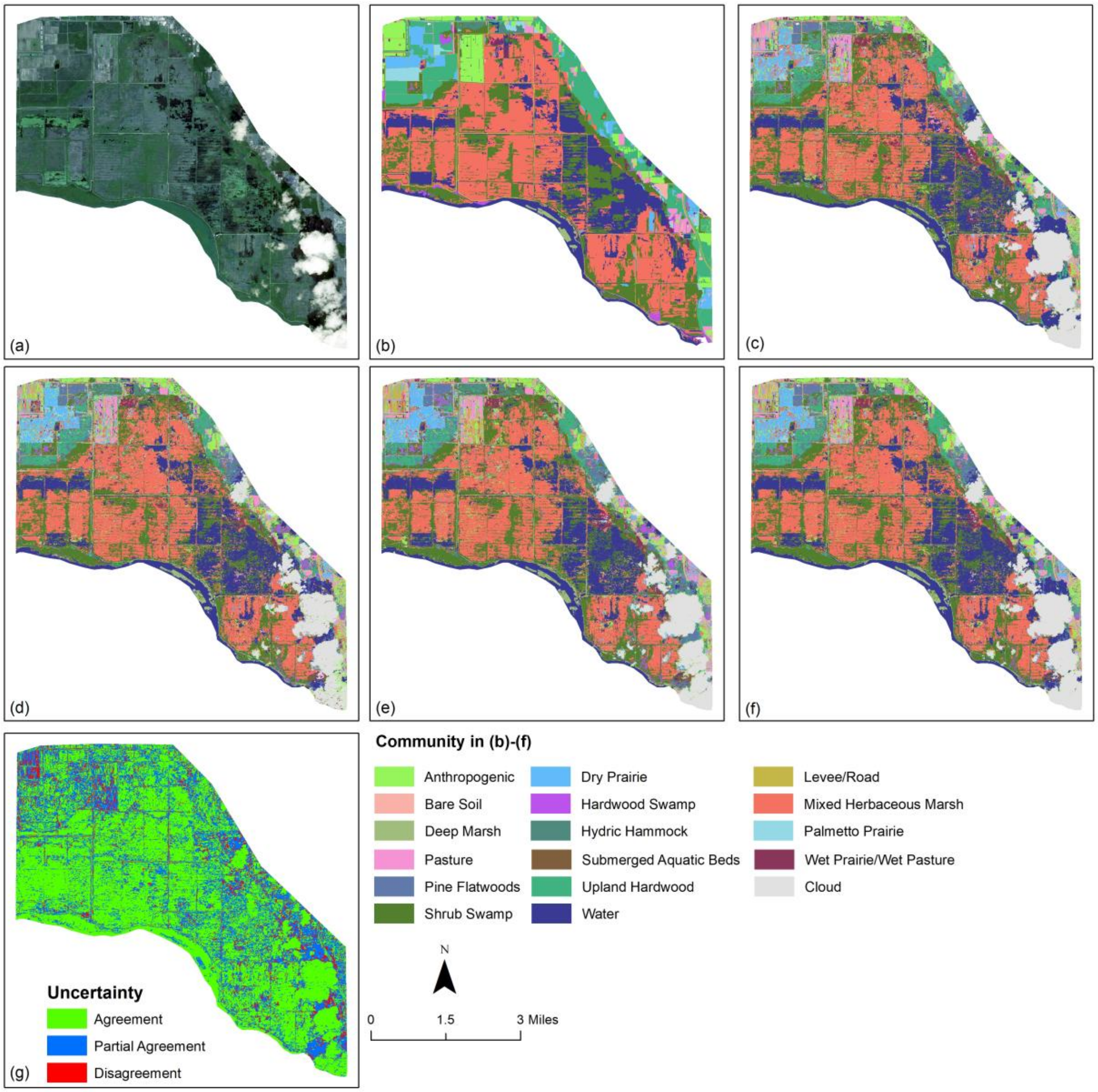


Figure 6 (a) WV-2 pansharpened imagery; (b) reference map from aerial photography; Object-based classifications from (c) SVM, (d) RF, (e) ANN, and (f) Ensemble analysis. (g) Uncertainty from ANN, SVM, and RF for east LANS.

4. Files for Quarter 3, 2021

Products in 2021 Quarter 3 are summarized in Table 4, and uploaded to District's ftp site.

Table 4. Products in Quarter 3, 2021 provided by FAU.

Areas	**Products**
OPRA	ArcGIS shapefiles of ANN, SVM, RF, Ensemble Analysis, Uncertainty; Excel file of error matrix
East LANS	
West LANS	

Appendix 1 Mapping Tutorial

Vegetation Mapping Using WorldView 2 Imagery, LiDAR DEM, and Object-based Machine Learning Classification

Caiyun Zhang

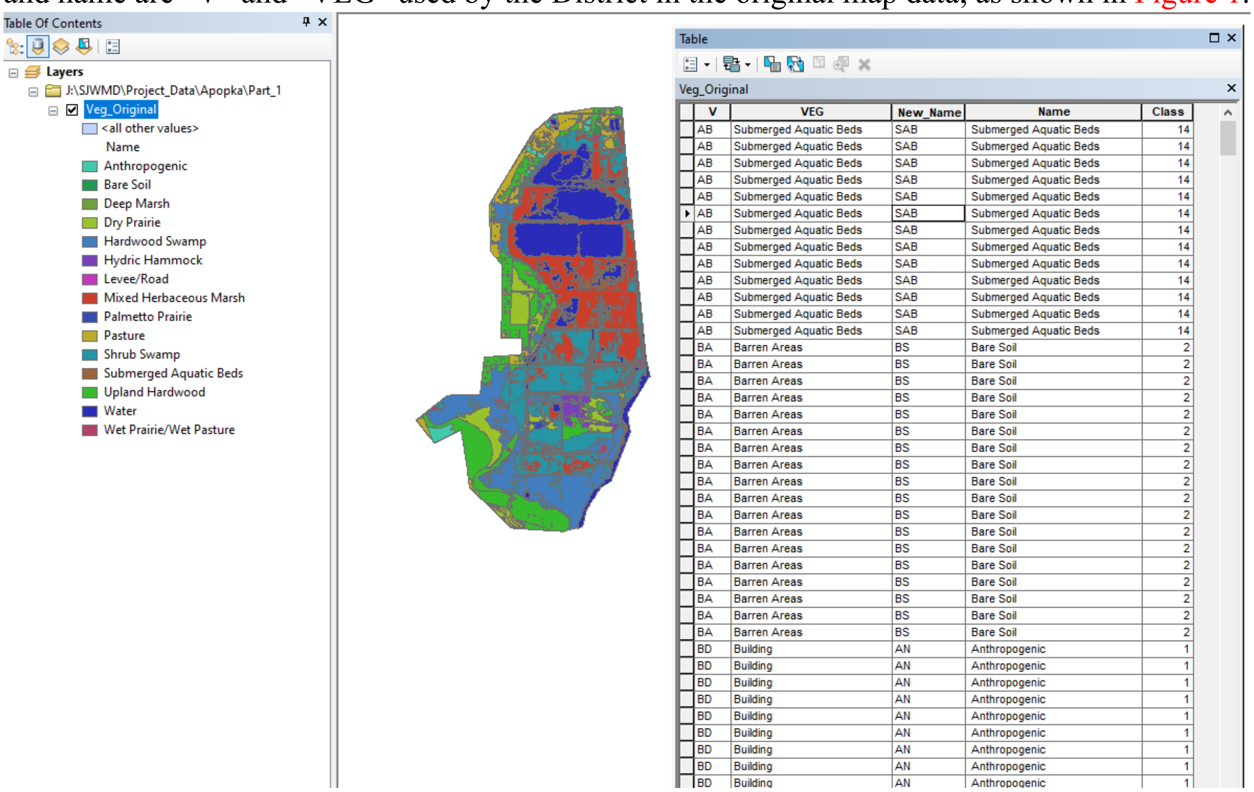
Florida Atlantic University

9/17/2021

This document describes the full tutorial for object-based machine learning classification and mapping of vegetation communities in the District wetlands using software packages ArcGIS, eCognition, and open source package WEKA. We use data from west Lake Apopka North Shore (LANS) as an example in this tutorial. ArcGIS skills are required.

Part I Identification of Communities to be Mapped

For a project area, based on the original plant community map data, we need to check which community's name need to modify to be consistent with the new names required by the District. This can be conducted in ArcGIS. A detailed tutorial of community name revision was provided in Quarter 1 report. Here, we added string type fields "New_Name" and "Name" to save the community abbreviations and names to be used for mapping. The old community abbreviations and name are "V" and "VEG" used by the District in the original map data, as shown in Figure 1.



V	VEG	New_Name	Name	Class
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
AB	Submerged Aquatic Beds	SAB	Submerged Aquatic Beds	14
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BA	Barren Areas	BS	Bare Soil	2
BD	Building	AN	Anthropogenic	1
BD	Building	AN	Anthropogenic	1
BD	Building	AN	Anthropogenic	1
BD	Building	AN	Anthropogenic	1
BD	Building	AN	Anthropogenic	1
BD	Building	AN	Anthropogenic	1
BD	Building	AN	Anthropogenic	1
BD	Building	AN	Anthropogenic	1
BD	Building	AN	Anthropogenic	1

Figure 1 Plant communities with revised names and corresponding abbreviations in the original map data. "V" and "VEG" are original names; "New_Name" and "Name" are added fields to save the revised names and abbreviations.

After name modification, we can find the total number of communities to be mapped. District also requires to detail areas labeled as "Upland" which may include Pines (Pine Flatwoods), crops (Anthropogenic), or other communities. These communities should also be included in mapping. The communities are finally encoded as numbers which are easier to use in the classification, as shown in Figure 1, an integer type field is added and we use a sequential number to represent each community as "1, 2, 3, …..". We create a text file "Veg_Code_WestApopka.txt" to save this. This file will be used later, as shown in Figure 2. For western Lake Apopka, we have a total of 17 classes to be identified. It is better to include very minor communities in the end in case we need to drop them in the classification if inadequate training samples are collected.

Veg_Code_WestApopka.txt.txt - Notepad

File Edit Format View Help

```
ClassCode, Community, Name
1, AN, Anthropogenic
2, BS, Bare Soil
3, DM, Deep Marsh
4, DP, Dry Prairie
5, HH, Hydric Hammock
6, HM, Mixed Herbaceous Marsh
7, HS, Hardwood Swamp
8, LR, Levee/Road
9, OW, Water
10, PA, Pasture
11, SS, Shrub Swamp
12, UH, Upland Hardwood
13, WP, Wet Prairie/Wet Pasture
14, SAB, Submerged Aquatic Beds
15, PP, Palmetto Prairie
16, PF, Pine Flatwoods
17, Cloud, Cloud
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

Figure 2 Text file to save the Class Code (numbers), Community, and Name to be used in the classification and mapping. Note here "Upland" is further classified into Pine Flatwoods, and Upland Hardwood. Cloud also appeared on the satellite imagery and was identified.

Part II Image Segmentation

To conduct object-based image classification, image segmentation is needed. We use Trimble eCognition Developer 10.1 for image segmentation. Steps are provided below.

- Start eCognition Developer 10.1, and click Create New Project . This step requires you to load in the imagery and we can navigate to the satellite imagery to be segmented, and name this project, as shown in Figure 3. Then Click OK.

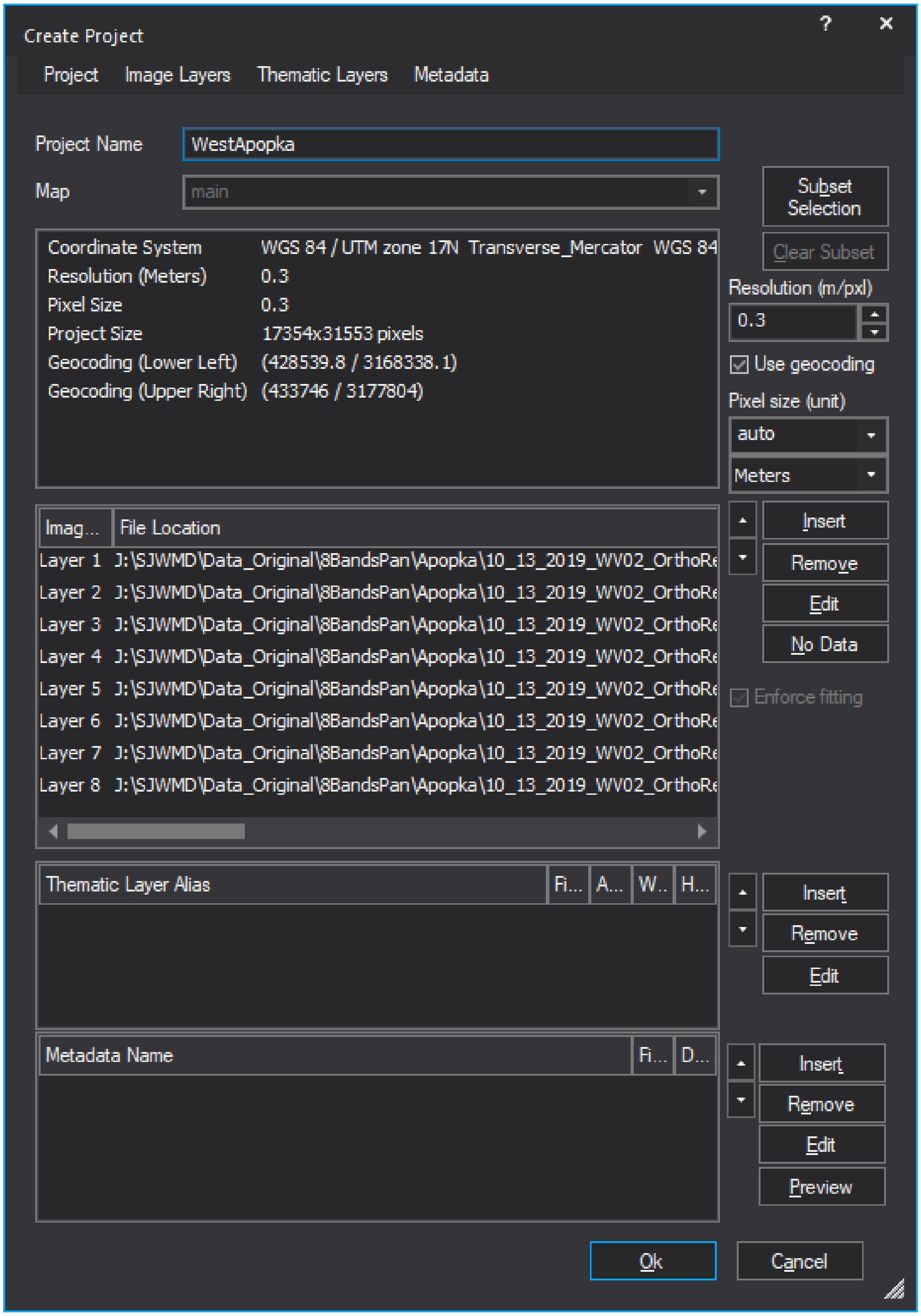


Figure 3 Create a new project and load in imagery in eCognition.

- There are 8 bands of WorldView-2 satellite imagery, as shown in Figure 3. In View Settings, we can change the imagery display as false color composite by using layers 7 (Near infrared), 5 (Red), and 3 (Green), as shown in Figure 4. This will enhance the imagery display. If View Settings does not appear in the main window, you can go to menu View->View Settings to dock it in the main window.

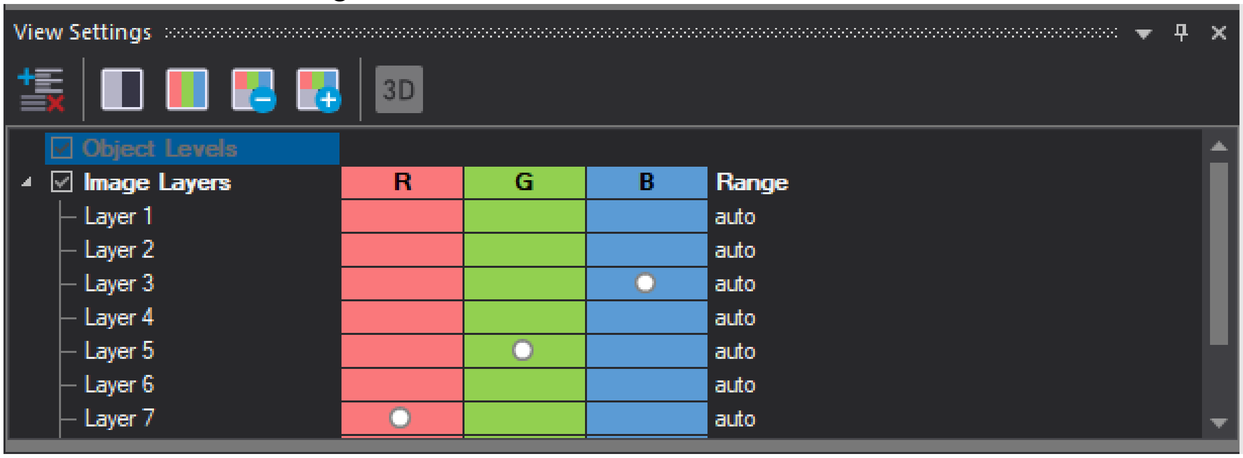


Figure 4 Image composite modification in eCognition.

- Multiresolution image segmentation: in Process Tree window, right click and select Add New Process, the Edit Process window will appear, as shown in Figure 5. In Algorithm select multiresolution segmentation, and algorithm parameters, set the Scale parameter to 150 (a small value will generate more objects, the dataset will be larger). Other parameters use default. Click Execute. Be patient, this will take a while, and maybe a few hours if large imagery is loaded.

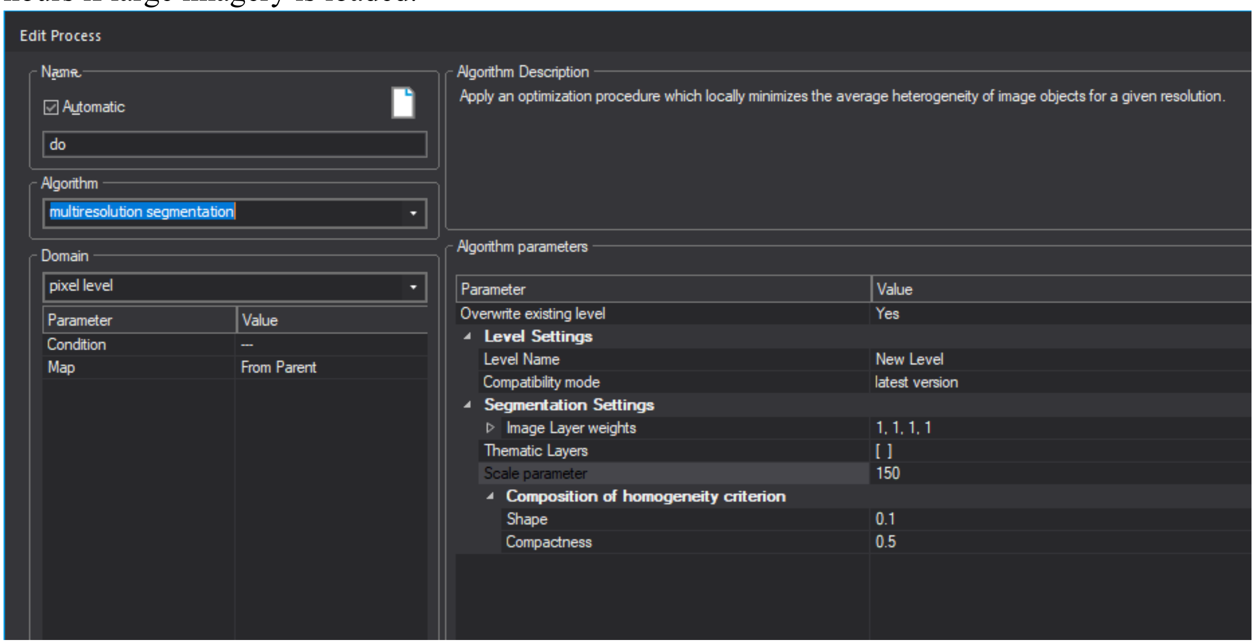


Figure 5 Edit process to set up multiresolution segmentation algorithm in eCognition.

- Export the segmentation results as ArcGIS shapefile: in Process Tree, right click and select Add New Process, the Edit Process window will appear. In Algorithm, select "export vector layer", in Algorithm parameters, set export path to where you want to save the shapefile; set Attributes to include layer mean, standard deviation, and texture attributes (GLCM) to be used in the classification, Shape Type to Polygon; Export Type: Smoothed; Export Format to Shapefile as shown in Figures 6-8. Click Execute. This may take more than 10 hours due to the large dataset.

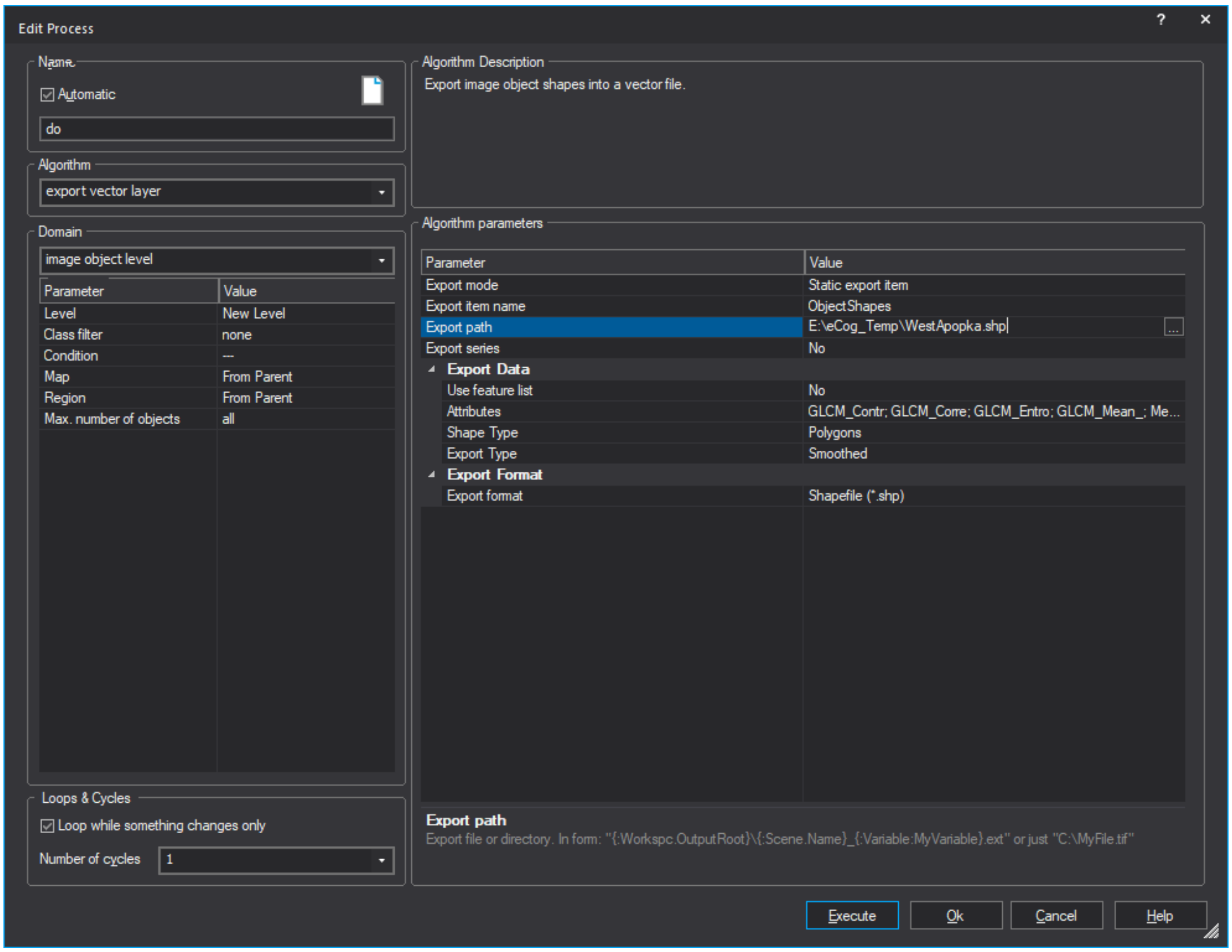


Figure 6 Export segmentation results to ArcGIS shapefile.

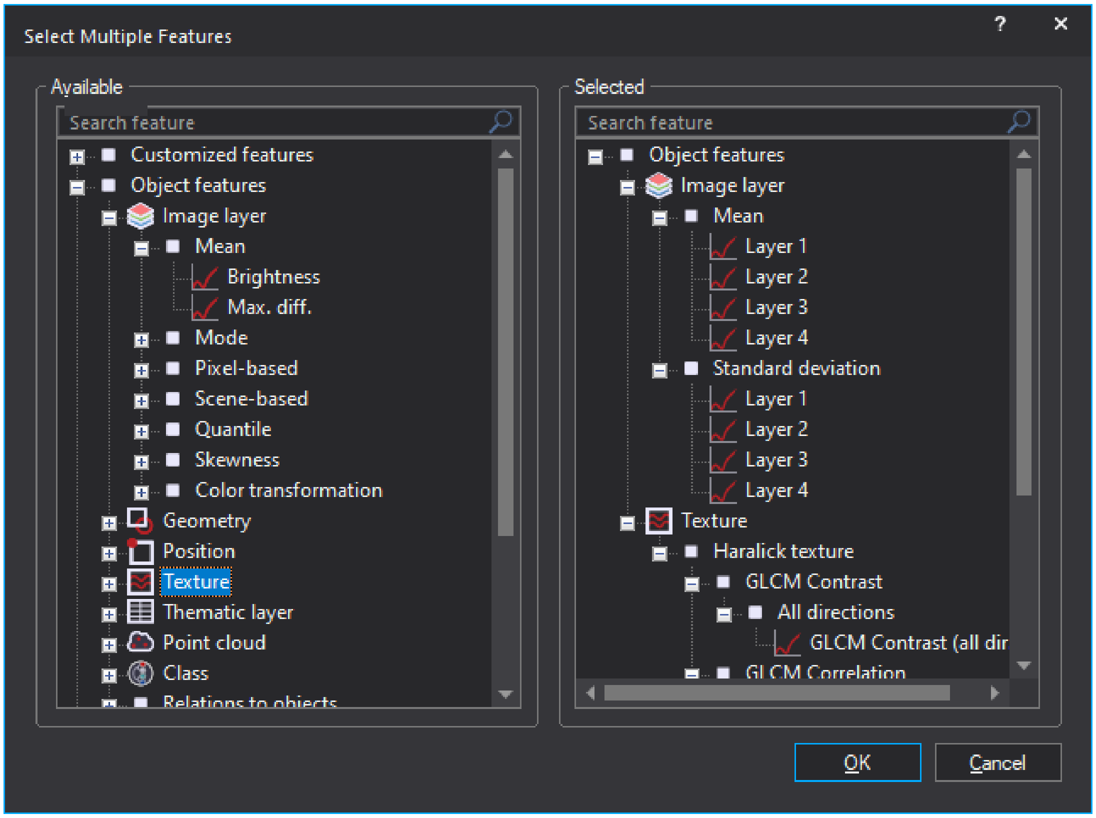


Figure 7 Select attributes to be exported.

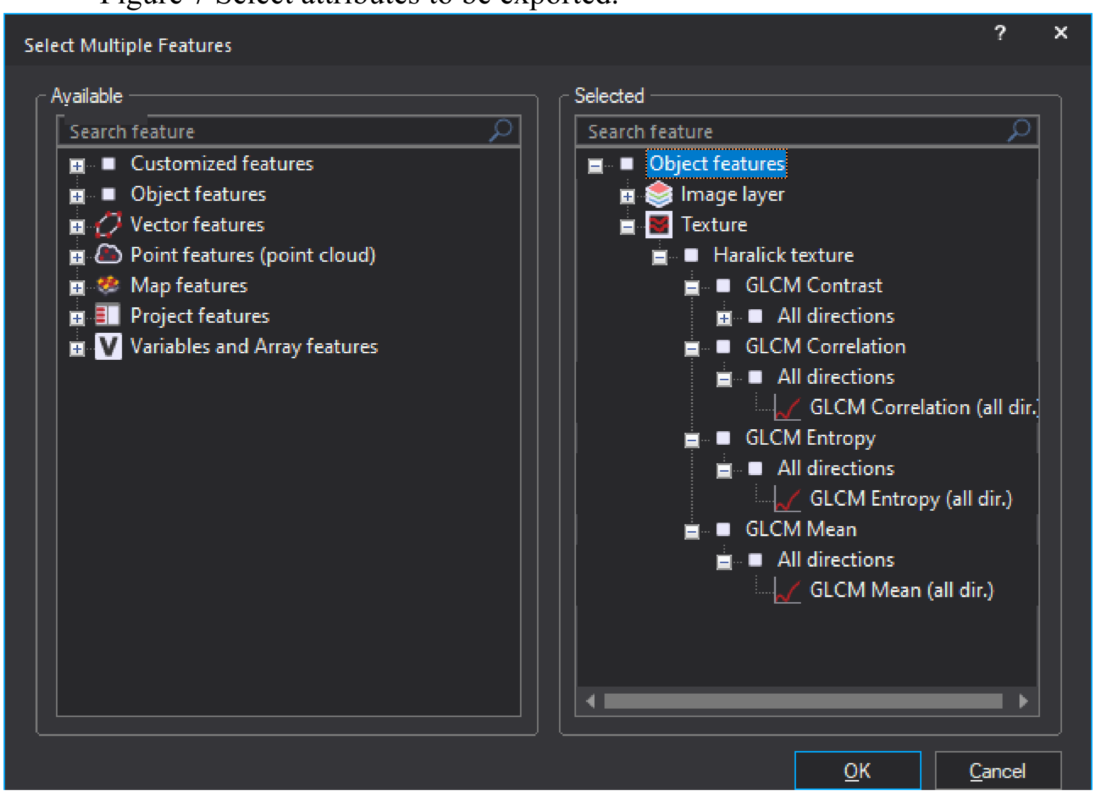


Figure 8 Select textures (a lot of them are correlated and here we only select including GLCM contrast, Entropy, and Mean all directions).

After export the vector segmentation file, we can close eCognition. Segmentation result for west LANS is displayed in ArcGIS in Figure 9.

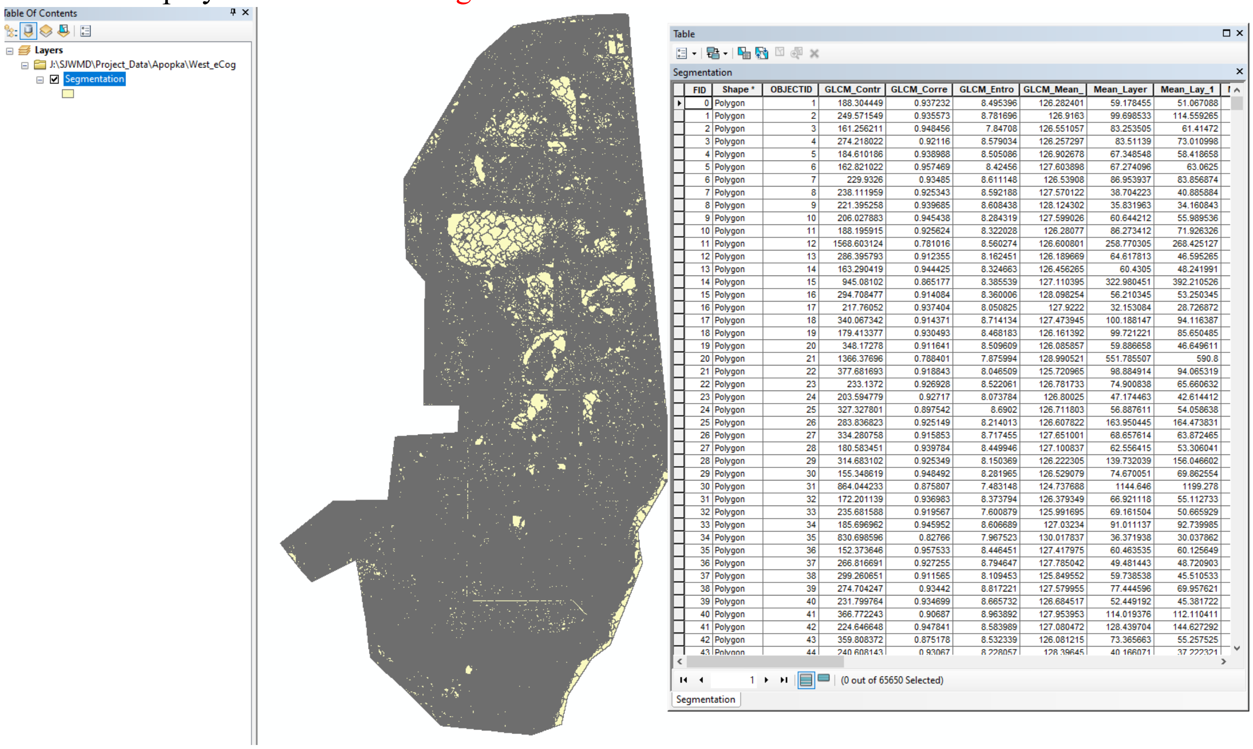


Figure 9 Exported segmentation ArcGIS shapefile with corresponding attributes.

Part III Add NDVI and LiDAR DEM Attributes

Normalized Difference Vegetation Index (NDVI) and/or other spectral indices might be useful for vegetation classification. LiDAR DEM may also help improve classification accuracy. We can use ArcGIS zonal statistics function to add these attributes in the classification. Steps are below.

- In ArcGIS, load in the segmentation shapefile file.
- Create four float type fields named as "NDVI_Mean" (NDVI mean), "NDVI_STD" (NDVI standard deviation), "DEM_Mean" (DEM mean), and "DEM_STD" (DEM standard deviation) to the segmentation file. These fields will save the NDVI and LiDAR DEM attributes.
- Load in the 8-band WV-2 imagery, using the imagery->Raster Functions->NDVI function to get the NDVI imagery. Ensure the visible band is set band 5, and infrared band is set to band 7, and the Scientific Output is checked. The result is displayed in Figure 10.
- Use the Zonal Statistics as Table function to generate the statistics of NDVI for each object, and the result will be a table, as shown in Figure 11.
- Join the statistical table with the segmentation file, and then calculate the NDVI_M and NDVI_STD two attributes from the statistical table using the mean and standard deviation.

- Similarly, load in the LiDAR DEM layer and conduct zonal statistics, and get the DEM_M and DEM_STD two attributes.

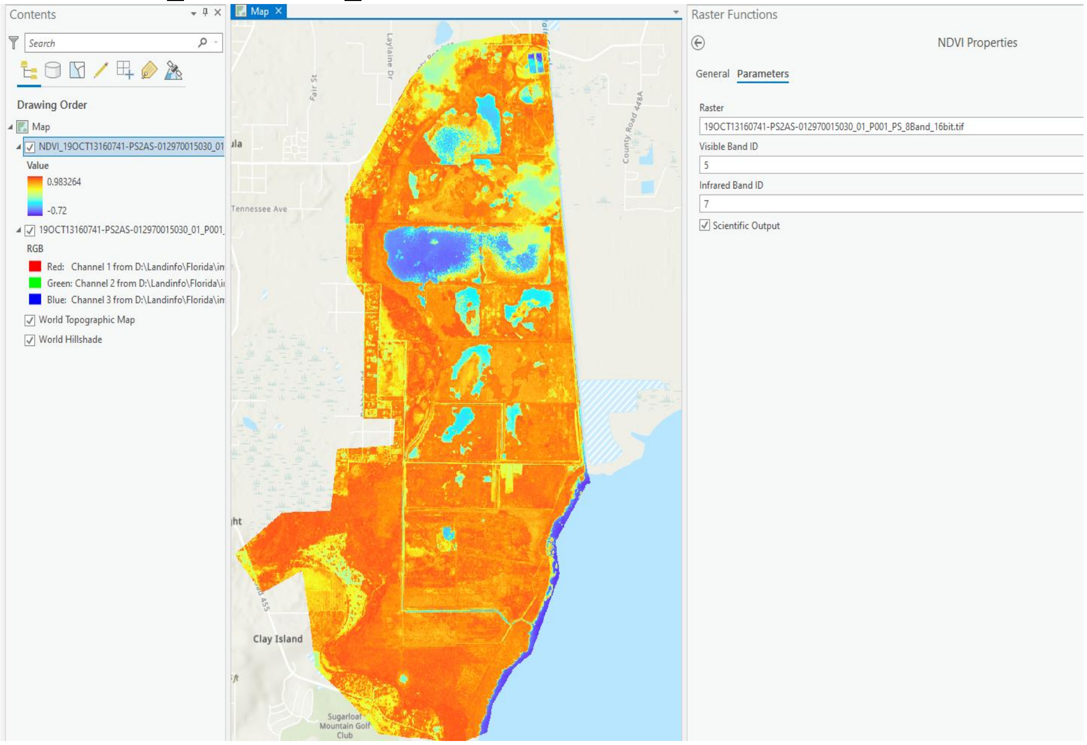


Figure 10 Derivation of NDVI imagery using the WV-2 imagery.

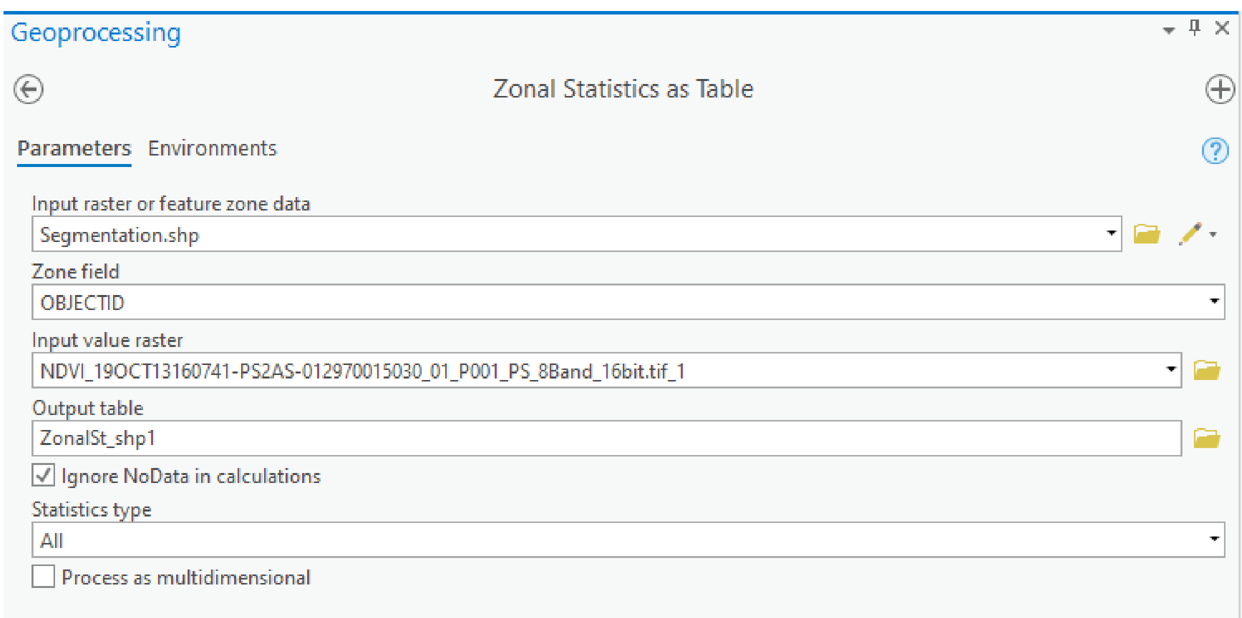


Figure 11 Generation of statistics of NDVI for each image object.

Part IV Training Sample Selection

Training samples are often selected from the image objects. Since the segmentation dataset has a large volume, it is challenging to manually select reference image objects directly from the segmentation file. We developed the point training sample selection approach to solve this issue. Steps are below.

- Create a new point vector shapefile in ArcGIS->View->Catalog Pane, and make sure the projection of this point layer is consistent with the current layers (i.e. the existing veg maps, using NAD_1983_HARN_UTM_Zone_17N), and then add an integer type new field "Class" in this point layer, which will be used to label the community.
- Load in this point dataset, original vegetation map dataset, corresponding satellite WV-2 imagery, and aerial photography used to create the old vegetation map.
- Edit this point layer in ArcGIS, and start adding points for each community based on the existing vegetation map (the one you have edited with new community names), and the satellite imagery. Highlight the community where you want to put training points and note its distribution and the coverage of this community, then start adding training points in Edit->Create. Make sure each point is within each community by changing each community symbol (the existing vegetation map) to be hollow, and checking the underlying imagery, to see whether each training sample point added looks like the intended community. After creating points for each community, label the points with the class code developed for each community as shown in Figure 3. Also make sure these training points are well spread over the AOI where the community is located. Figure 12 displays an example of the training points.

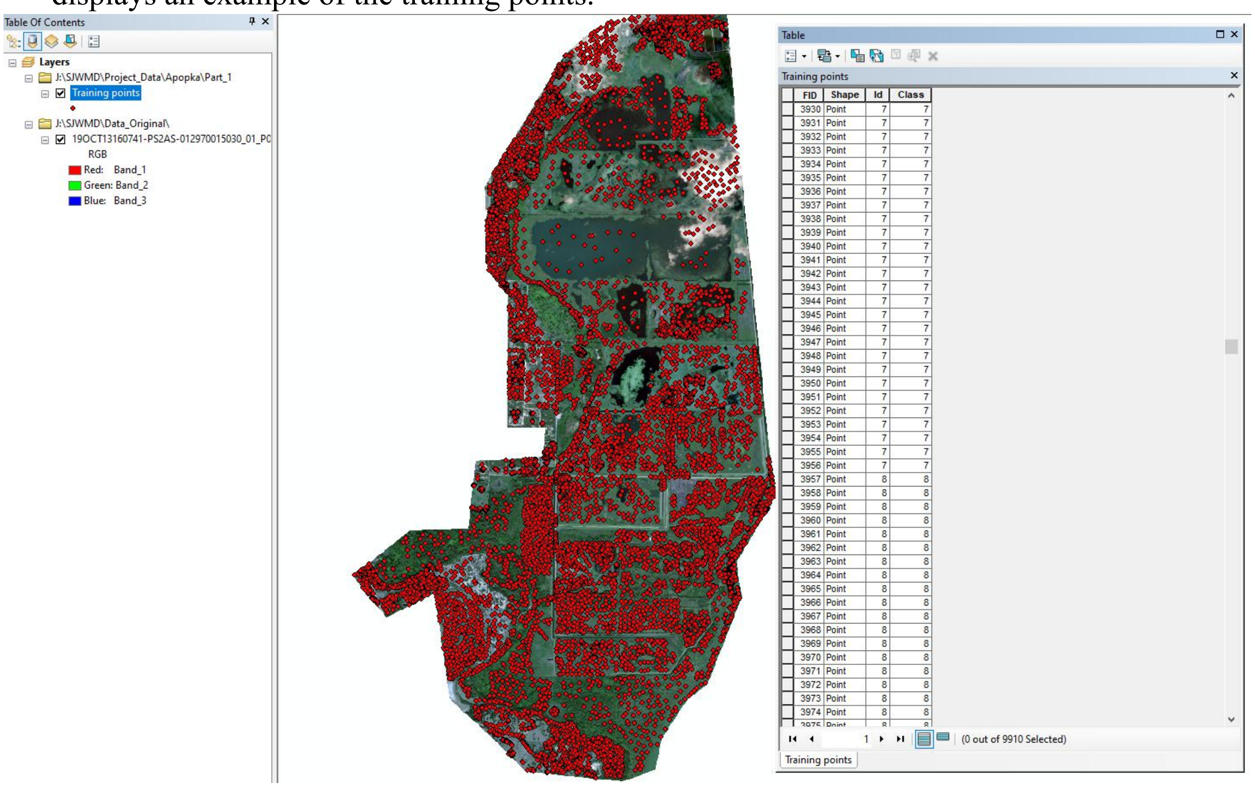


Figure 12 Labeled training points to be used for collecting training data.

- After editing the point training data layer, the point layer can then be spatially joined with the segmentation polygon shapefile, and then exported. The Avg_Class field will be in the joined polygon file. The default value is 0, polygons with non-zero value are the training object polygons. In this way, the training objects/segments can be identified, selected and exported as csv file as training data. In the export step, make sure the attributes of other point features will not be exported. Only export eCognition exported attributes, NDVI, and LiDAR attributes and the class code with non-zero values.
- The interface for the spatial join between the point layer and segmentation polygon layer is displayed in Figure 13, and the final training objects are shown in Figure 14. The exported csv text file is displayed in Figure 15.

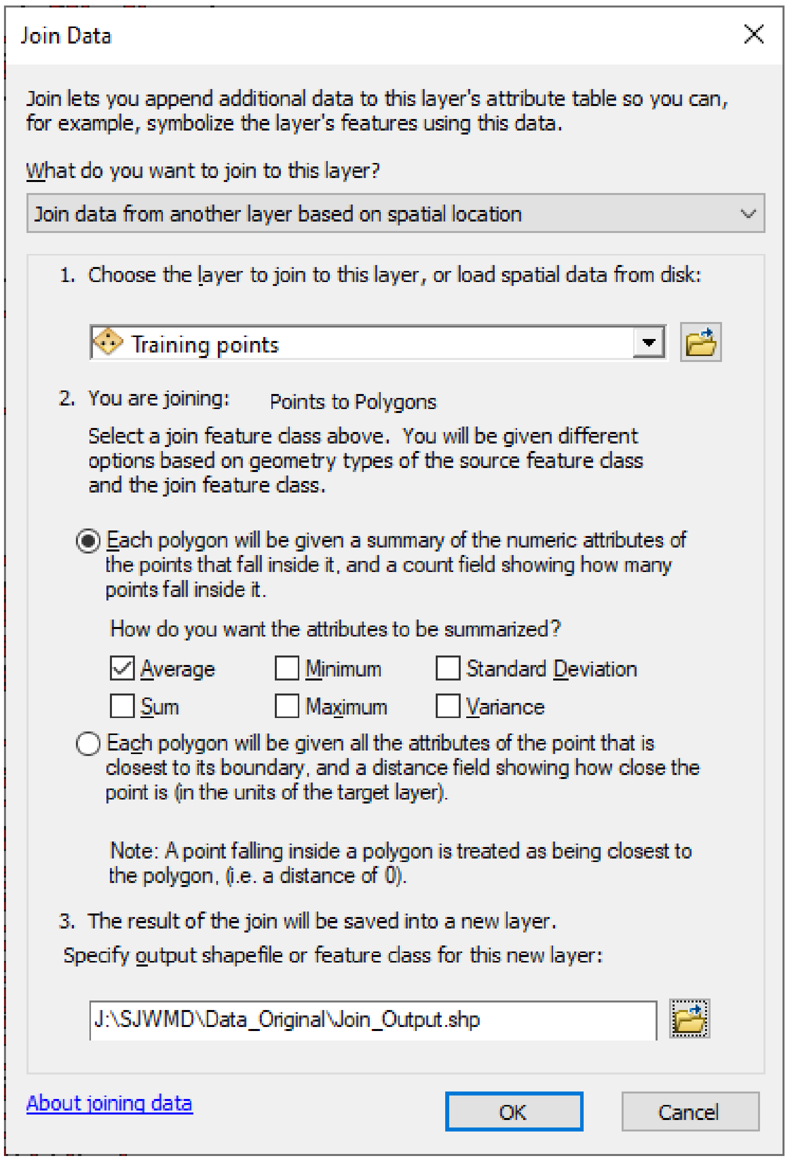


Figure 13 Spatially join the segmentation polygon file with the labeled point layer to identify the training polygons.

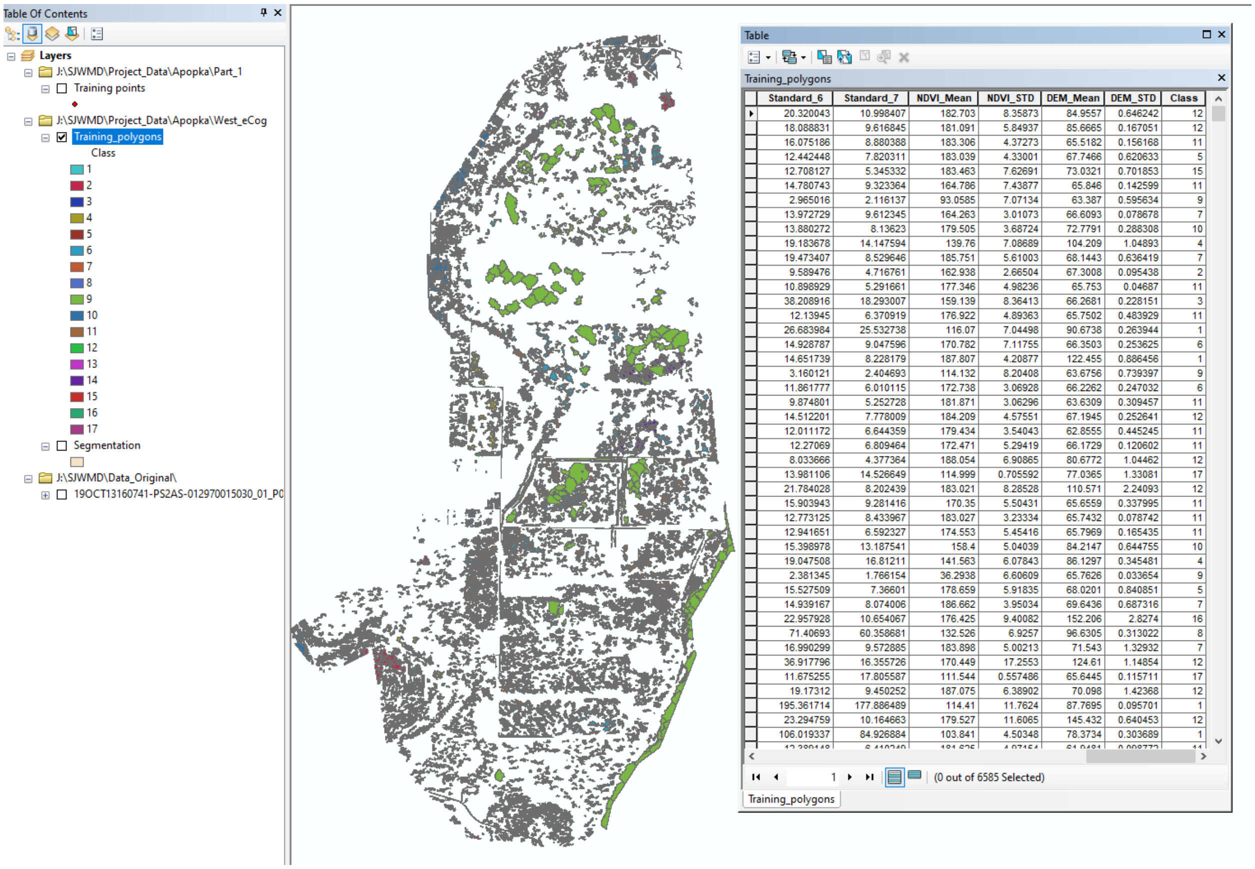


Figure 14 Identified training objects/polygons and its corresponding attribute table.

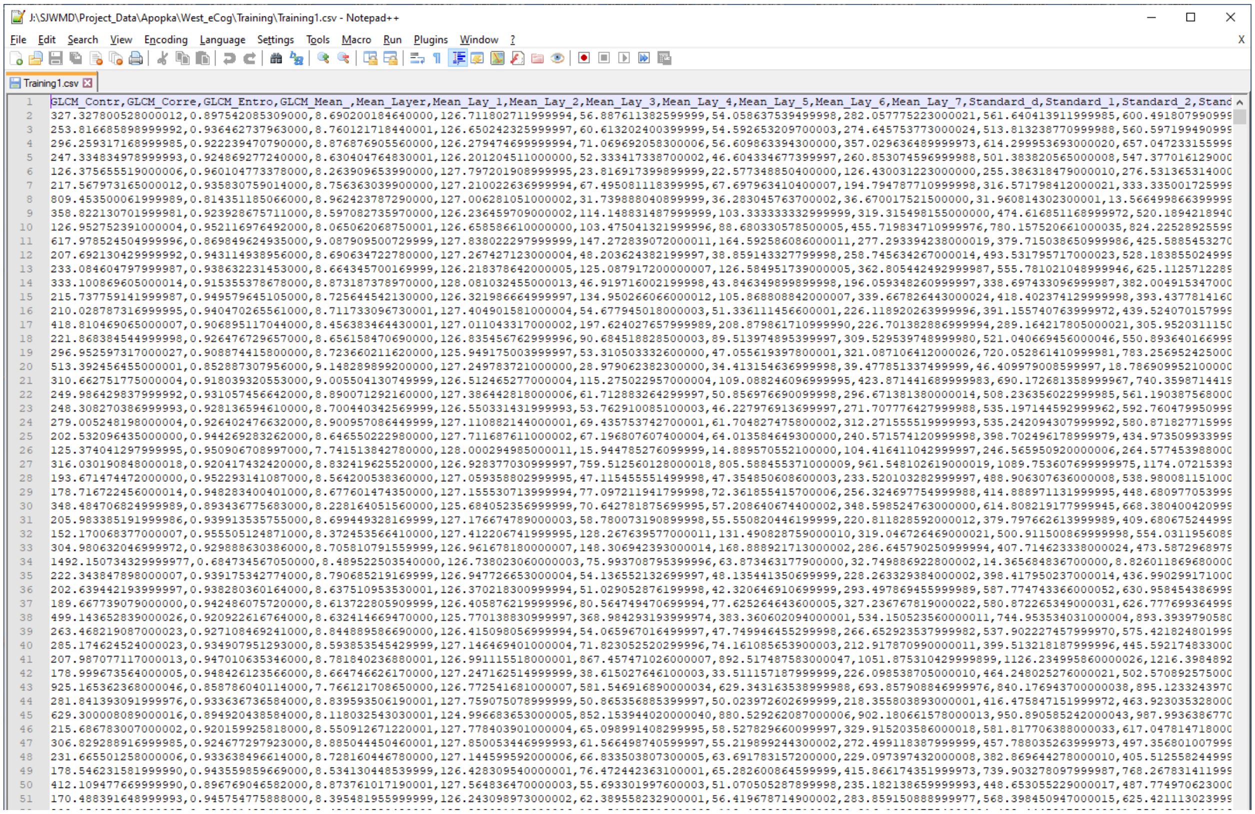


Figure 15 Exported training data in csv text file from ArcGIS (attributes of each object.)

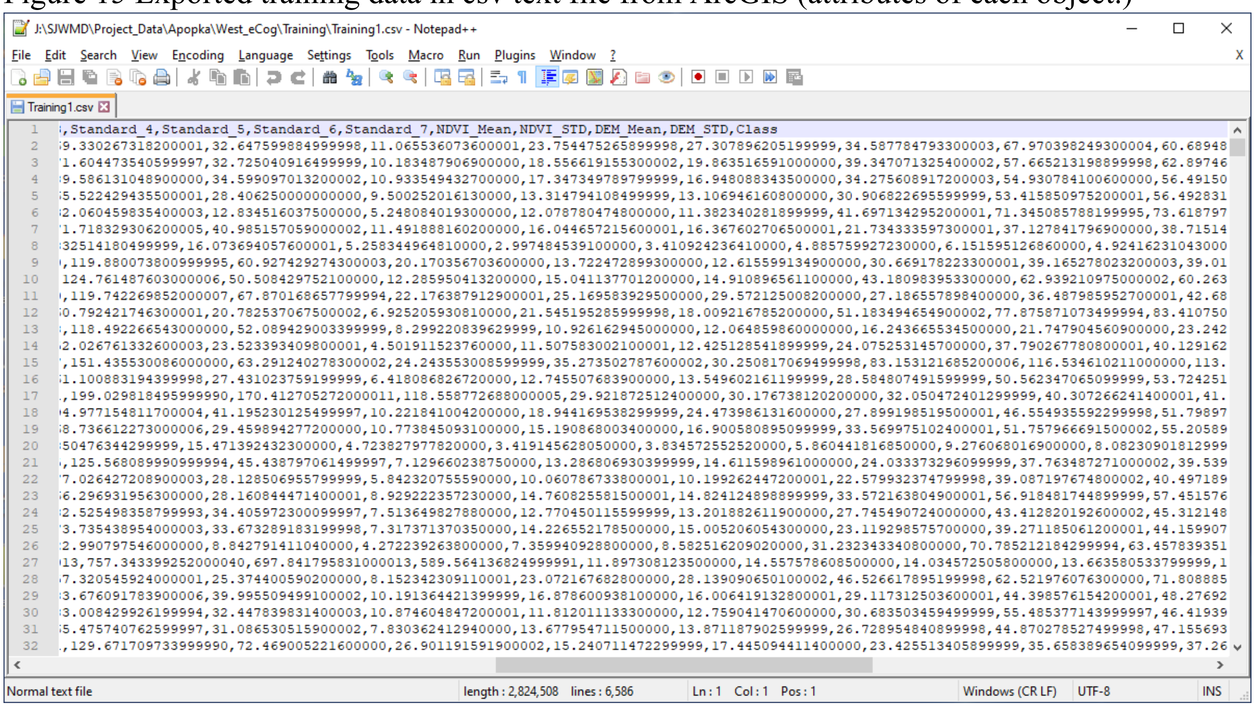


Figure 15 Continued. Note last attribute is the class attribute to save class code of the training samples. It is an integer corresponding Figure 2 (the text file we created for vegetation community coding)

Part V Machine Learning Model Development

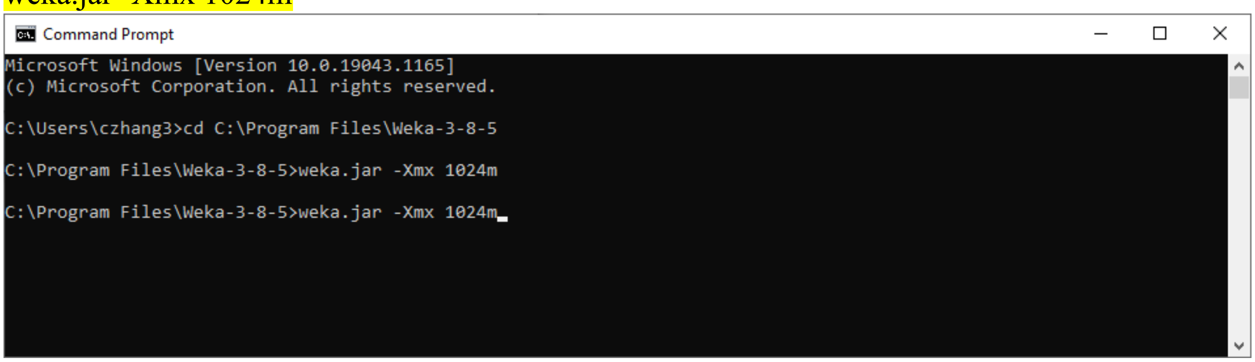
We use the exported training data in csv file to develop a machine learning model. Here we use WEKA, an open source machine learning package (https://www.cs.waikato.ac.nz/ml/weka/index.html) to conduct this major step. WEKA contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization. Steps are below.

- Download WEKA from https://www.cs.waikato.ac.nz/ml/weka/index.html or https://sourceforge.net/projects/weka/ and install the stable version for Windows.
- WEKA tutorial and relevant documents can be found from https://waikato.github.io/weka-wiki/documentation/
- Increase WEKA data processing memory. The default memory is small and easy to crash when process large datasets like ours. To increase WEKA memory, start Command Prompt and go to the path (e.g. my WEKA is installed in "C:\Program Files\Weka-3-8-5") where WEKA is installed, type in command (after type in click enter) as below:

  cd C:\Program Files\Weka-3-8-5

  Then type in
  weka.jar -Xmx 1024m

  WEKA will start automatically and get into the Explorer interface with an increased memory to 1024 mb.
- WEKA Explorer is a GUI environment for exploring data. The Weka GUI Chooser has menus like Program to set up log window, Tools, and Visualization. The function of each menu is detailed in WEKA tutorial. We will use Explorer to complete our task.
- In Weka Explorer's Preprocess, click Open file… and load in the csv training data we exported from ArcGIS, as shown in Figure 16 (ensure File Type is CSV data file, and the file extension name is .csv). Figure 17 shows the training data load in interface.
- Save the training data as ARFF file by clicking the save button. WEKA processes and develops models using ARFF (Attribute-Relation File Format). ARFF is an ASCII text

file that describes a list of instances sharing a set of attributes. Details of ARFF and organization of ARFF can be found in WEKA tutorial. The saved ARFF of training data is displayed in Figure 18.

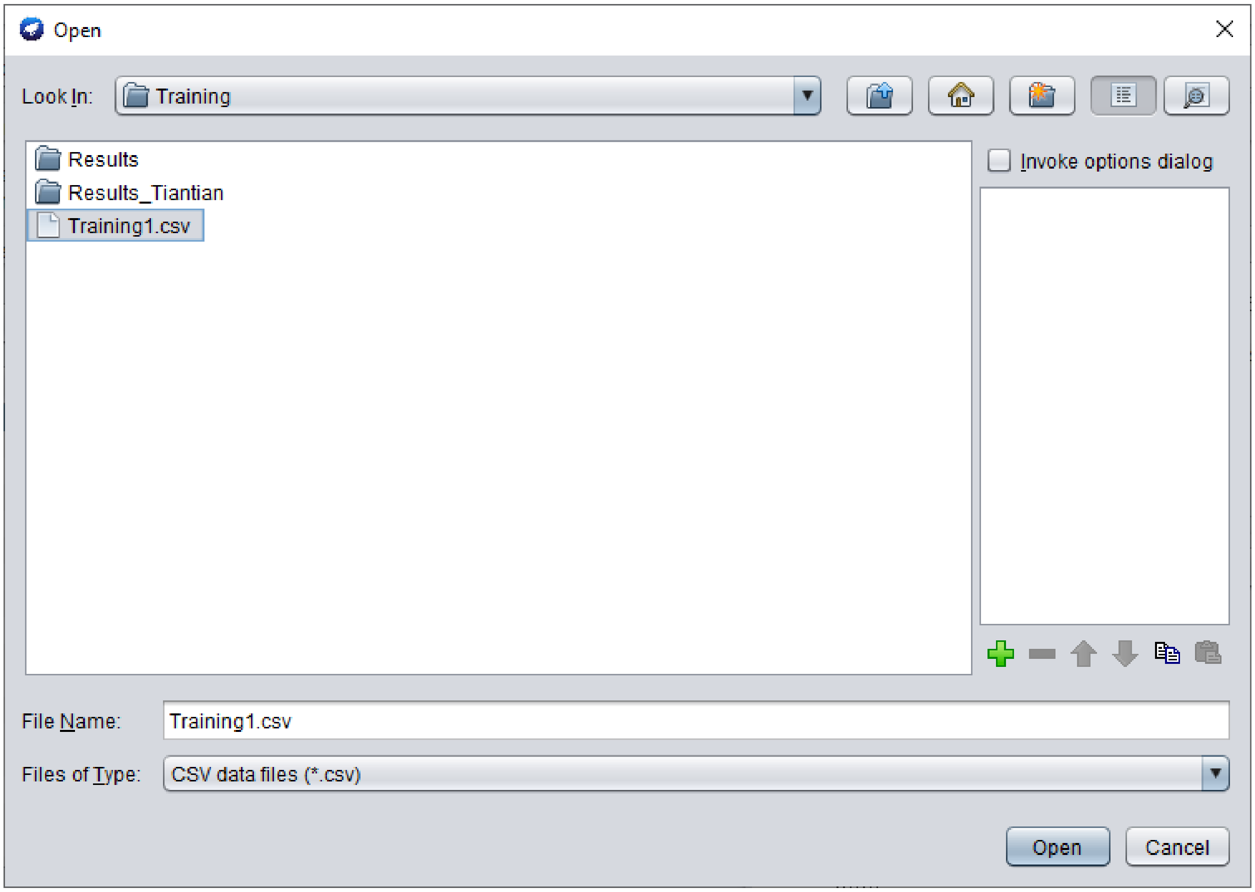


Figure 16 Training csv text file load in WEKA (note the Files Type, and file extension name is lower letter csv).

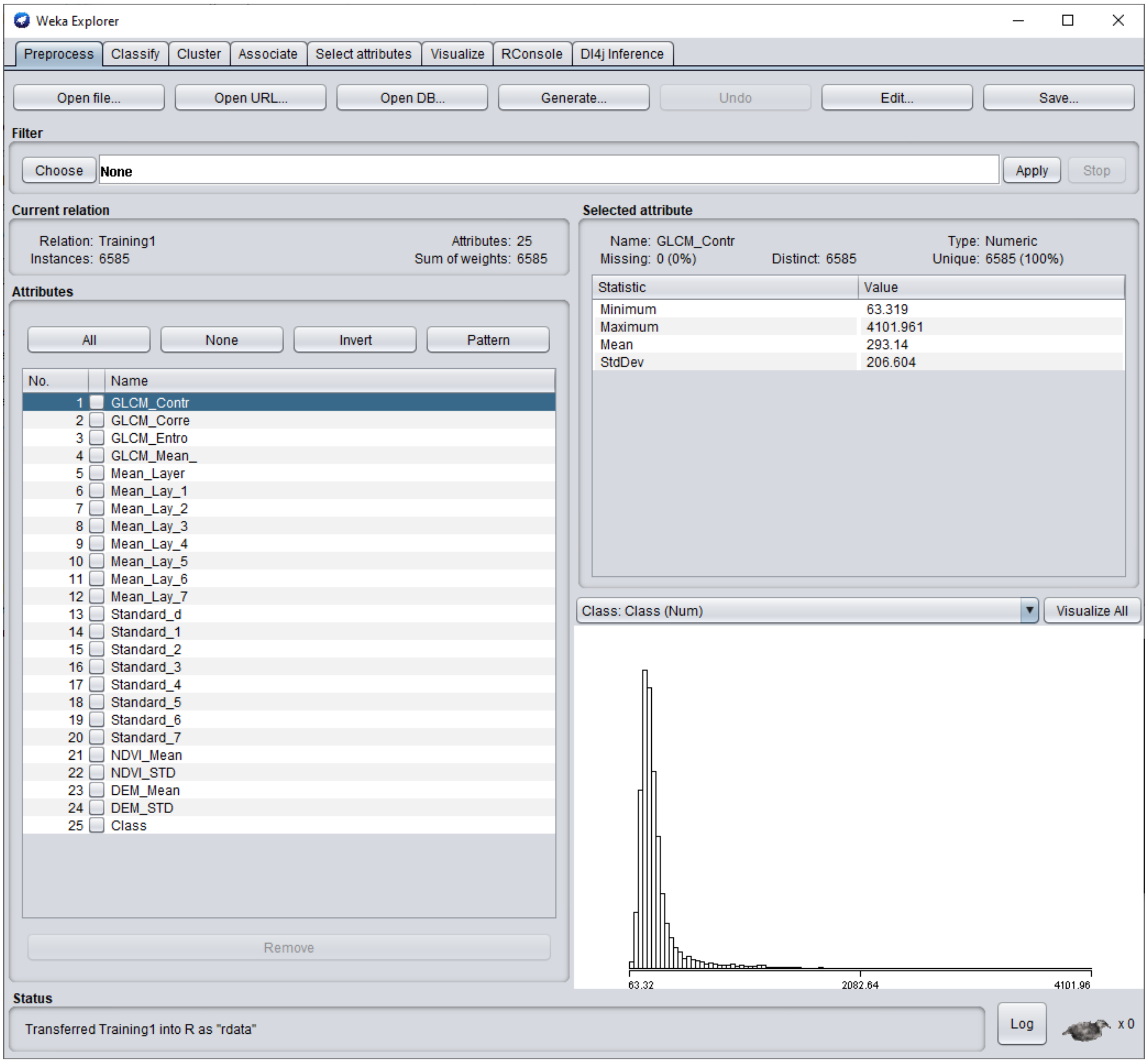


Figure 17 Load in training csv file in WEKA. If unexpected attributes were exported from ArcGIS, here you can check these attribute and click the Remove button.

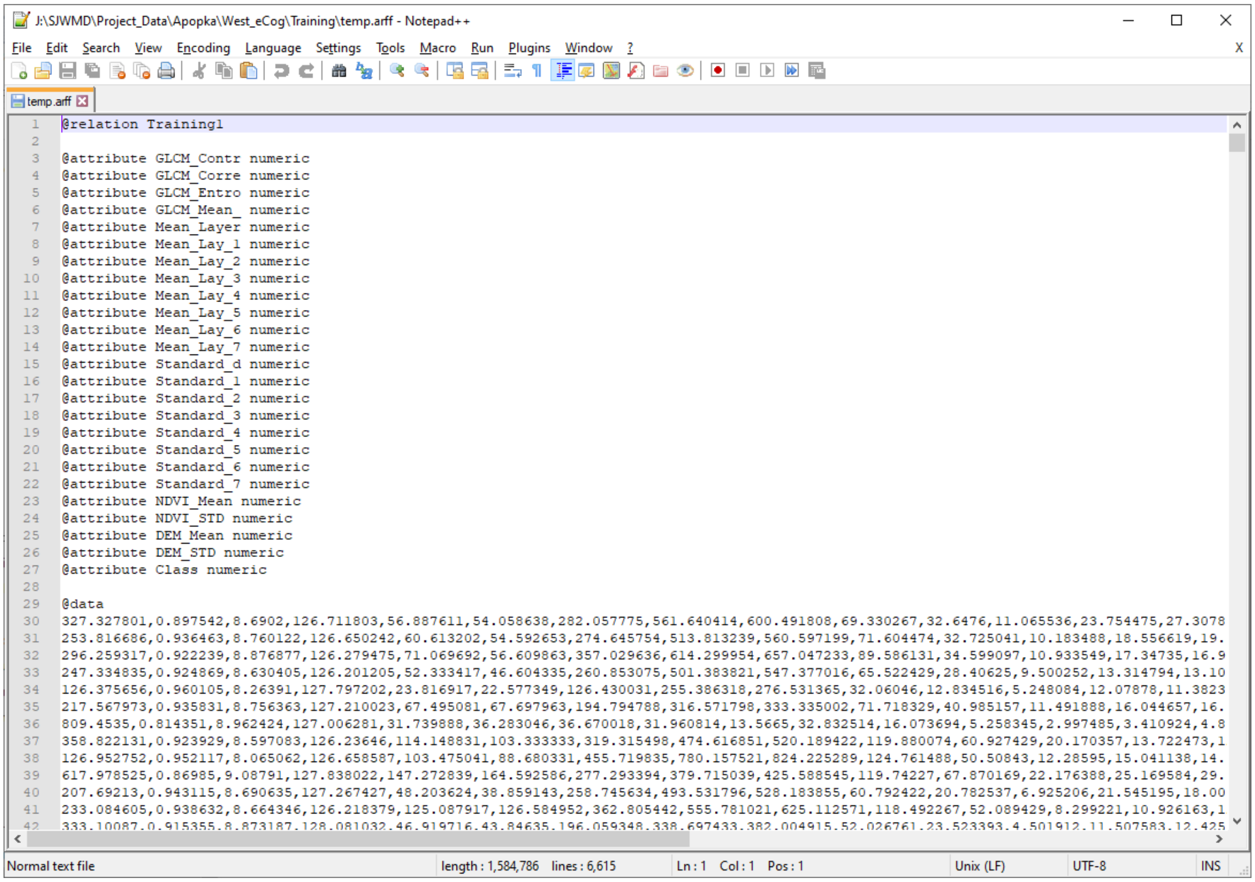


Figure 18 Saved ARFF of training data from WEKA. Note that all attributes are numerical.

- Revise the ARFF. In WEKA, attributes can be numerical (a real or integer number), string (a list of characters), or nominal (a predefined list of values). Since we will do classification, rather than regression, we need to revise the last attribute Class into nominal. To do this, we can open this ARFF text file using notepad++ (https://notepad-plus-plus.org/downloads/), and revise:

  @attribute Class numeric- > @attribute Class {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17}

  Note here we have 17 classes to classify (refer to Figure 2, and each class is encoded into numbers). After revision, the ARFF looks like Figure 19.

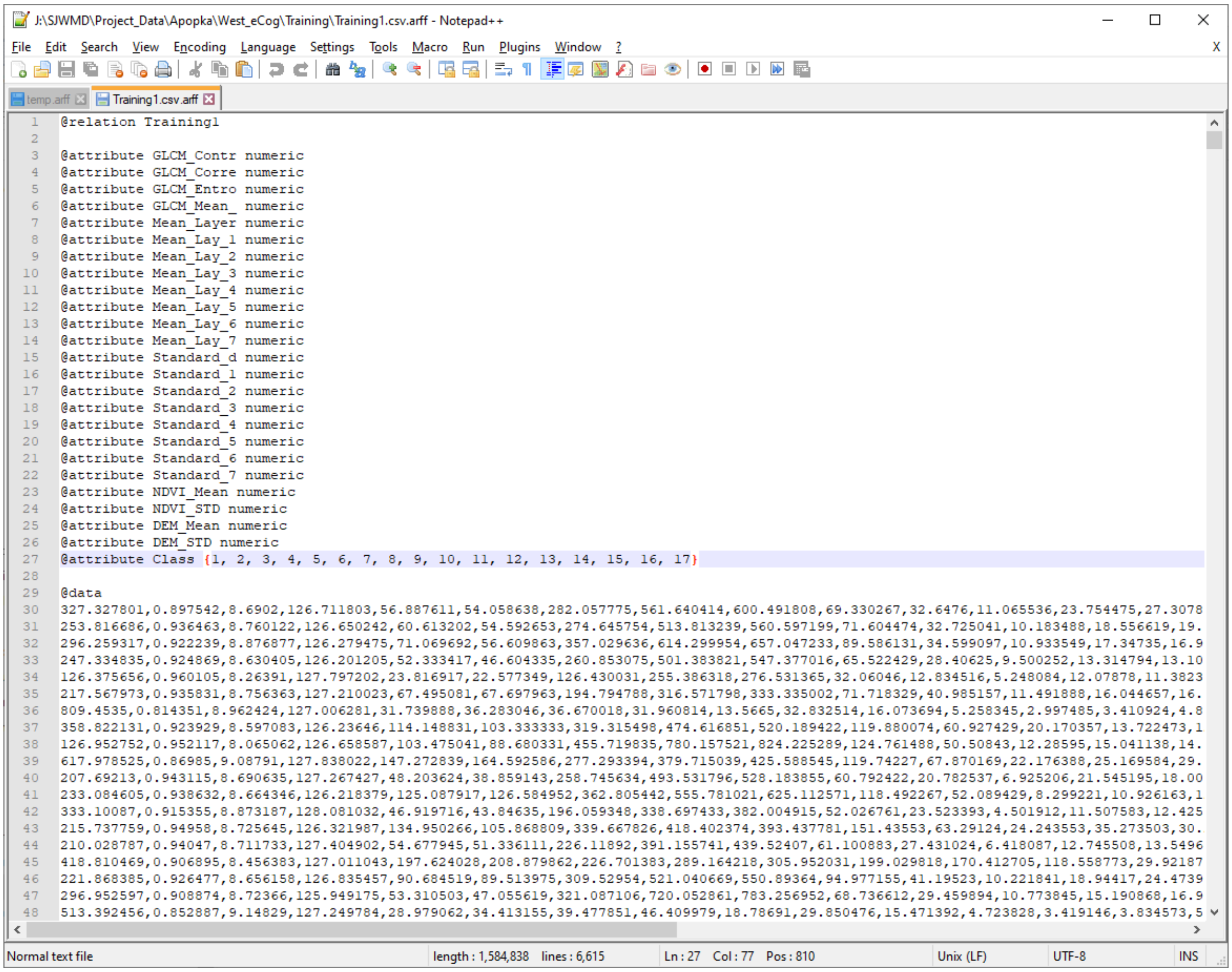


Figure 19 Revised ARFF ready to develop model.

- Load in the revised ARFF file in WEKA: use Open File…, and now the Files of Type should be Arff data files (*arff). If we check the Class attribute in Attributes, we can see this attribute is now Nominal type, and the visualization shows the total number of training samples for each class, as shown in Figure 20.

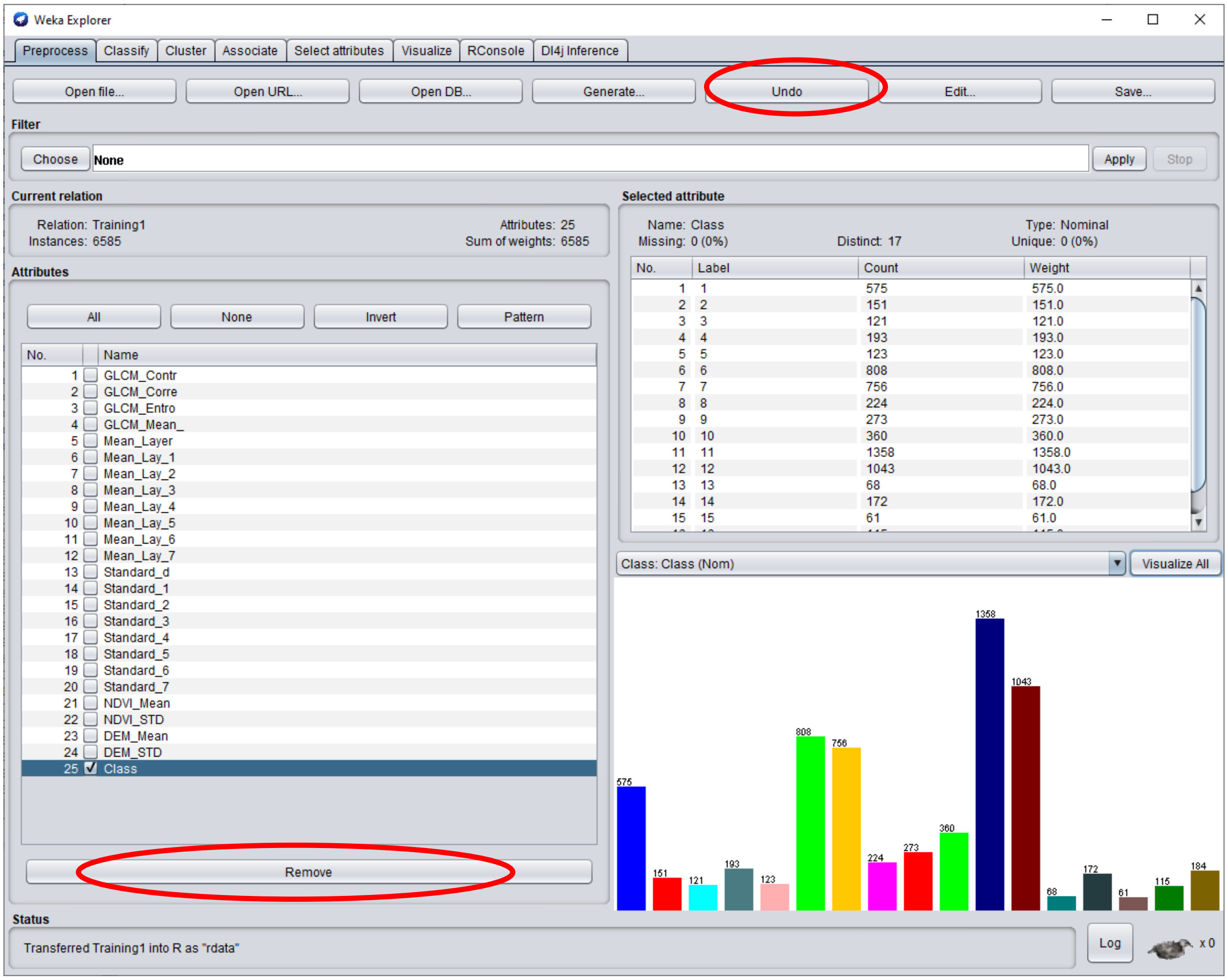


Figure 20 Revised Arff training data file. Again, here if you see unexpected attributes, you can select these attributes and click the Remove button to drop them in the training process. If you accidently dropped some attributes, you can click the Undo button to call them back.

- Now data process is done and we can develop model using the Classify tab.
- Click the Classify tab, in Classifier, click Choose, we can see a lot of classifiers available in WEKA. Here we will test Artificial Neural Network (ANN), Random Forest (RF), Support Vector Machine (SVM), and k Nearest Neighbor (kNN) algorithms.
- Let us work on ANN first: Choose->functions->MultilayerPerceptron. This is the commonly used Multiple Layer Perceptron (MLP) neural network classifier. Left Click the selected classifier in the text box next to Choose, the GenericObjectEditor dialog box will appear as shown in Figure 21. This is the place we can adjust input parameters required by each algorithm. There are three critical parameters in MLP: hiddenLayers (the number of hidden layers), Learning Rate (the learning rate), and trainingTime (the total cycles the MLP). They are highlighted in Figure 21. We should use a smaller learning rate like 0.1 (the default is 0.3), and set the training time to 300 to avoid over training. If we click the More button, more information about the selected classifier and the meaning of each parameter will appear like Figure 22. After set up the classifier, Click OK.

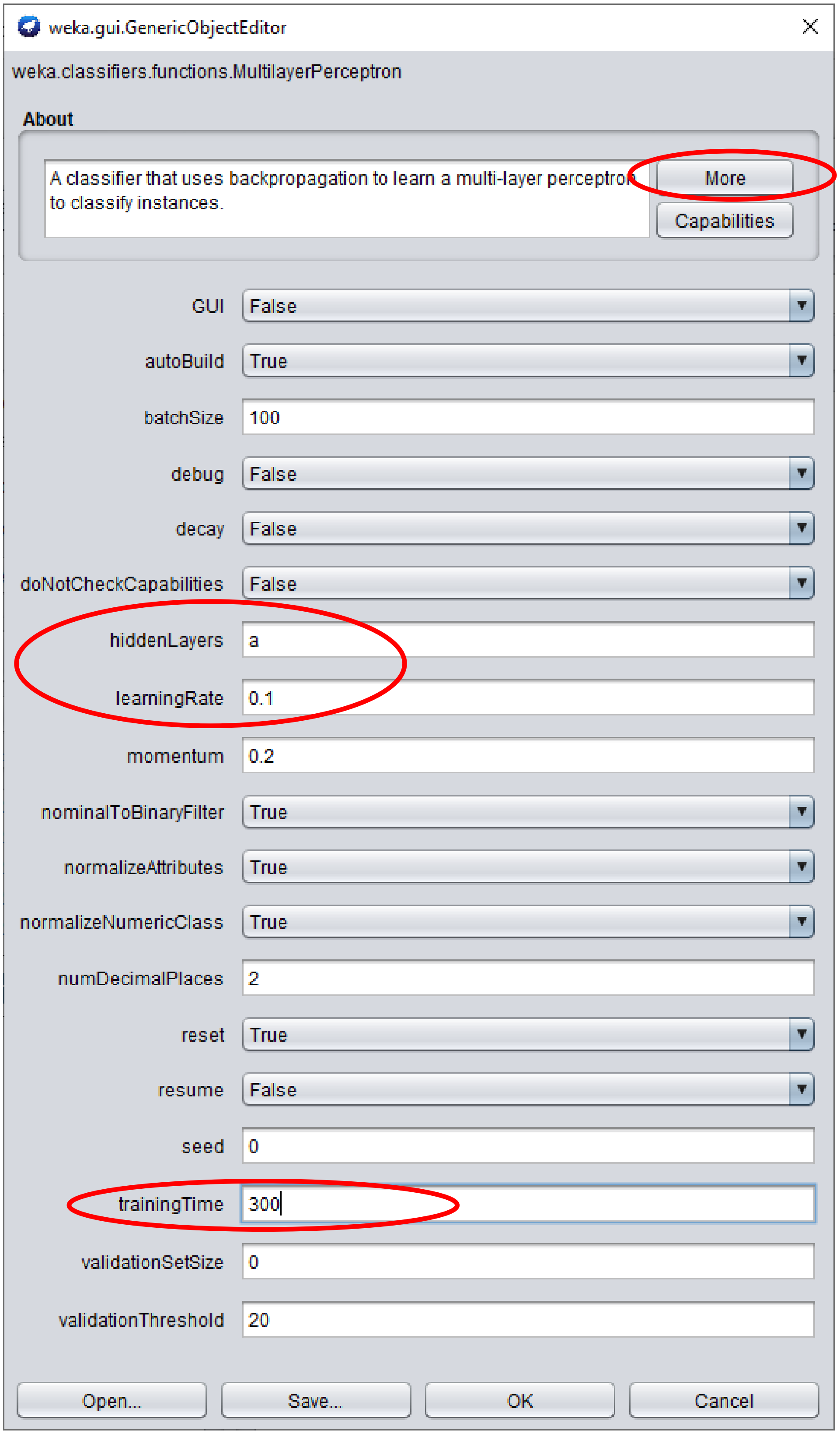
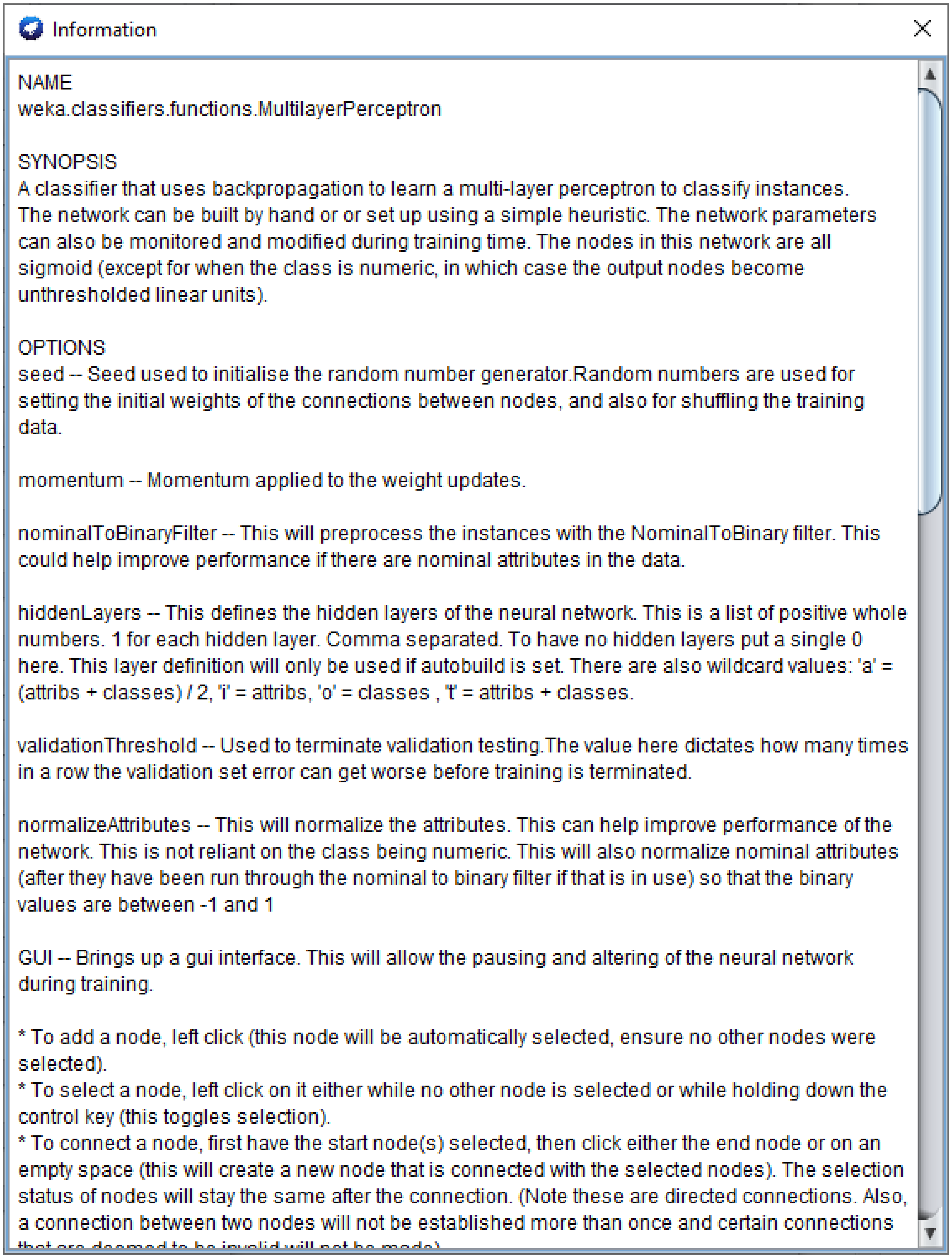


Figure 21 Interface of MLP ANN in WEKA.



Information

NAME
weka.classifiers.functions.MultilayerPerceptron

SYNOPSIS
A classifier that uses backpropagation to learn a multi-layer perceptron to classify instances. The network can be built by hand or or set up using a simple heuristic. The network parameters can also be monitored and modified during training time. The nodes in this network are all sigmoid (except for when the class is numeric, in which case the output nodes become unthresholded linear units).

OPTIONS
seed -- Seed used to initialise the random number generator.Random numbers are used for setting the initial weights of the connections between nodes, and also for shuffling the training data.

momentum -- Momentum applied to the weight updates.

nominalToBinaryFilter -- This will preprocess the instances with the NominalToBinary filter. This could help improve performance if there are nominal attributes in the data.

hiddenLayers -- This defines the hidden layers of the neural network. This is a list of positive whole numbers. 1 for each hidden layer. Comma separated. To have no hidden layers put a single 0 here. This layer definition will only be used if autobuild is set. There are also wildcard values: 'a' = (attribs + classes) / 2, 'i' = attribs, 'o' = classes , 't' = attribs + classes.

validationThreshold -- Used to terminate validation testing.The value here dictates how many times in a row the validation set error can get worse before training is terminated.

normalizeAttributes -- This will normalize the attributes. This can help improve performance of the network. This is not reliant on the class being numeric. This will also normalize nominal attributes (after they have been run through the nominal to binary filter if that is in use) so that the binary values are between -1 and 1

GUI -- Brings up a gui interface. This will allow the pausing and altering of the neural network during training.

* To add a node, left click (this node will be automatically selected, ensure no other nodes were selected).
* To select a node, left click on it either while no other node is selected or while holding down the control key (this toggles selection).
* To connect a node, first have the start node(s) selected, then click either the end node or on an empty space (this will create a new node that is connected with the selected nodes). The selection status of nodes will stay the same after the connection. (Note these are directed connections. Also, a connection between two nodes will not be established more than once and certain connections

Figure 22 Detailed explanation of the selected MLP classifier.

- After set up the classifier, Click More options on the Weka Explorer interface as shown in Figure 23. This brings up the setup of output selection, as shown in Figure 24. Choose CSV, and click Evaluation metrics as shown in Figure 25. Then Click OK.

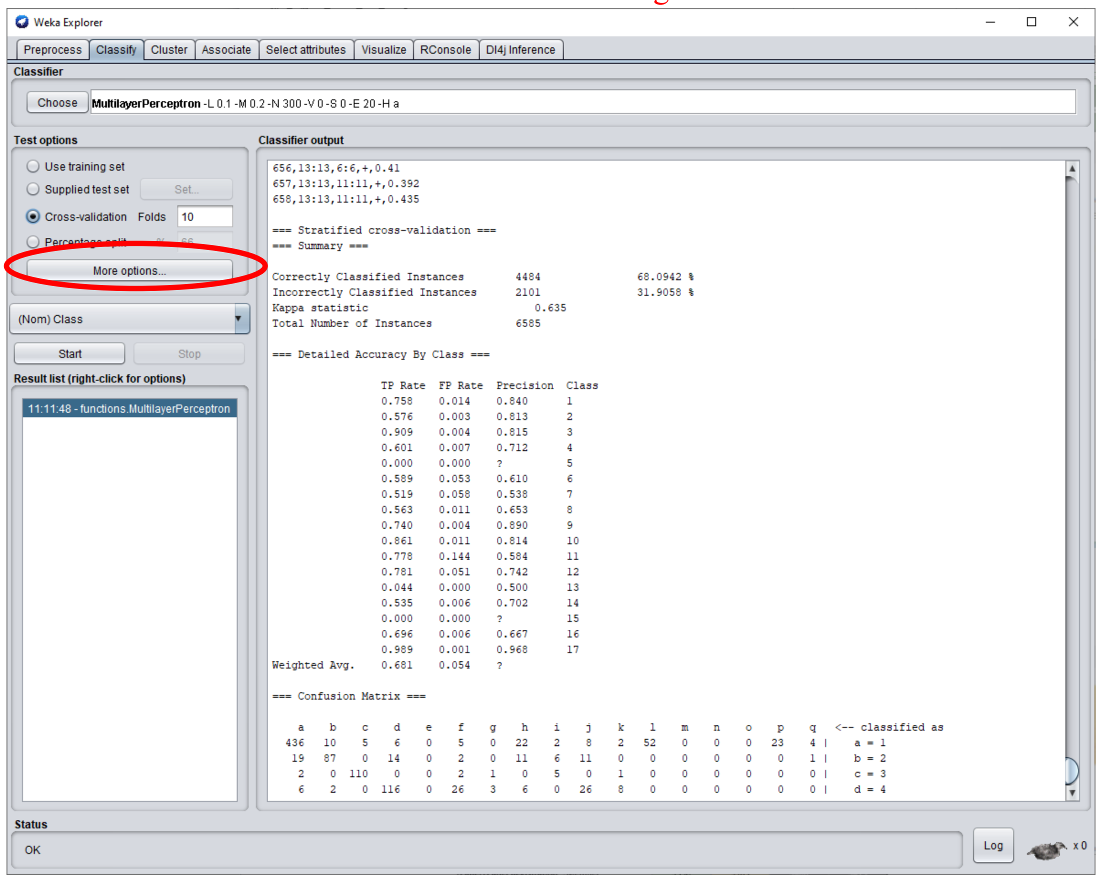


Figure 23 Setup of other parameters.

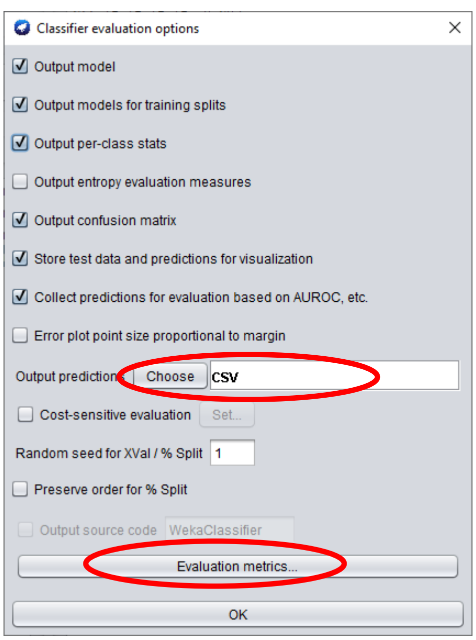


Figure 24 Setup of outputs.

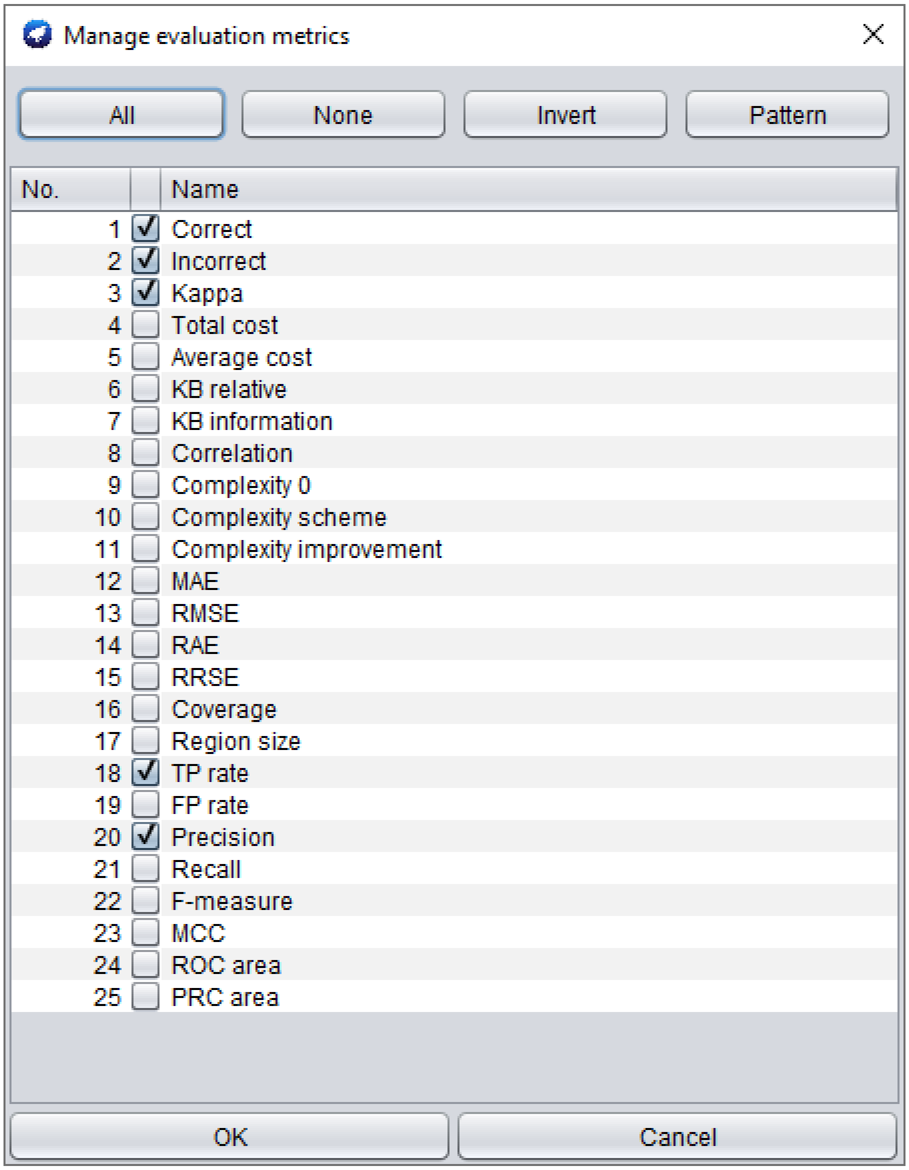


Figure 25 Setup of evaluation metrics. We only select Corrected, Incorrect, Kappa, TP Rate (User's accuracy), and Precision (Producer's accuracy).

- Test Options: this is to set up the test modes including using training set (The classifier is evaluated on how well it predicts the class of the instances it was trained on); Supplied test set (The classifier is evaluated on how well it predicts the class of a set of instances loaded from a file. Clicking the Set... button brings up a dialog allowing you to choose the file to test on); Cross-validation (The classifier is evaluated by cross-validation, using the number of folds that are entered in the Folds text field), and Cross-validation (The classifier is evaluated by cross-validation, using the number of folds that are entered in the Folds text field). We will use cross-validation options. More options can be set by clicking More options…

- Select (Nom) Class as the attribute to be classified. Then Click Start. WEKA will run the classifier and the bird will be moving around and stop when the classification is completed. The output will be in the text box of Classifier output as shown in Figure 26. Left click the output, and use Ctrl+A to select all, and Ctrl+C to copy, and then paste the output to a new txt file to save the output of the classification. You can also automatically save the output to a csv file by setting up in More options.

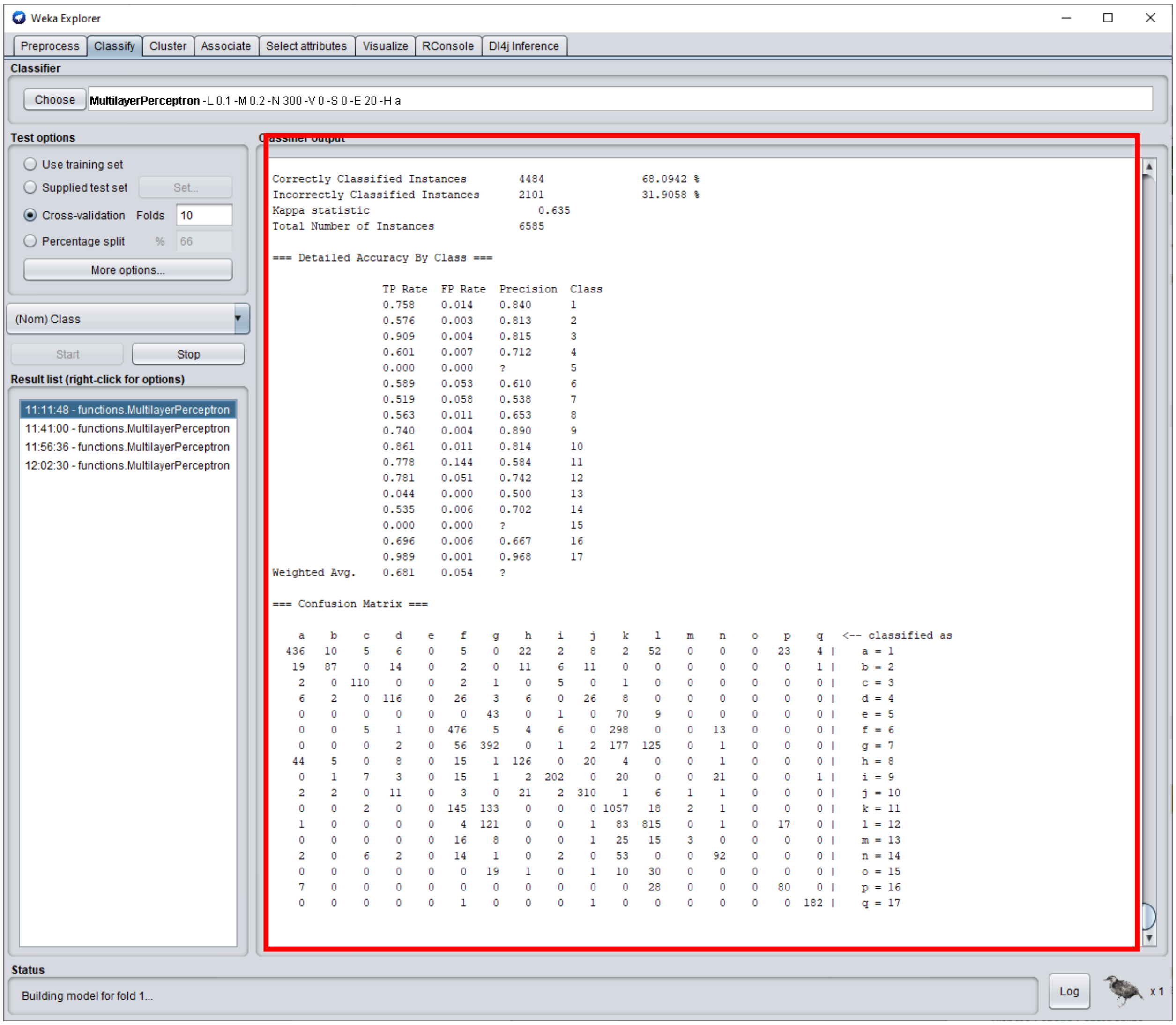


Figure 26 Classified output: Correctly classified instances (overall accuracy), incorrectly classifier instances (error), Kappa statistics (Kappa value), TP, FP, and Precision of each class, and error matrix.

- The result list: left clicking can save the model, view in different window, delete model, etc. When you run multiple times of a model, you may need to delete the results to release the memory.
- Similarly, we can run the SVM classifier by clicking Choose functions->SMO. SMO refers to the specific efficient optimization algorithm used inside the SVM implementation, which

stands for Sequential Minimal Optimization. There are two critical parameters in configuring SVM, C parameter, called the complexity parameter which controls how flexible the process for drawing the line to separate the classes can be. A value of 0 allows no violations of the margin, whereas the default is 1. The other key parameter in SVM is the type of Kernel to use. The simplest kernel is a Linear kernel that separates data with a straight line or hyperplane. The default in Weka is a Polynomial Kernel that will separate the classes using a curved or wiggly line, the higher the polynomial, the more wiggly (the exponent value). A popular and powerful kernel is the RBF Kernel or Radial Basis Function Kernel that is capable of learning closed polygons and complex shapes to separate the classes. It is a good idea to try a suite of different kernels and C (complexity) values on your problem and see what works best. Here we choose the RBF Kernel (Figure 27). By left clicking the text box of selected RBF kernel, it will bring up the configuration of the RBF kernel (Figure 28). We can set the gamma to 5.0.

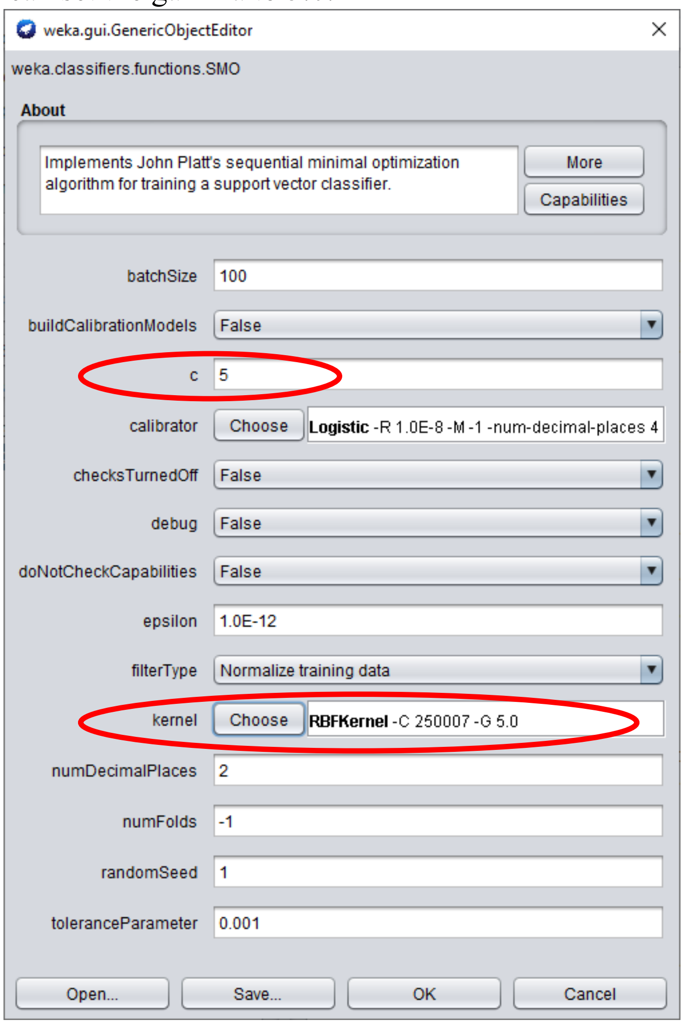


Figure 27 SVM configuration.

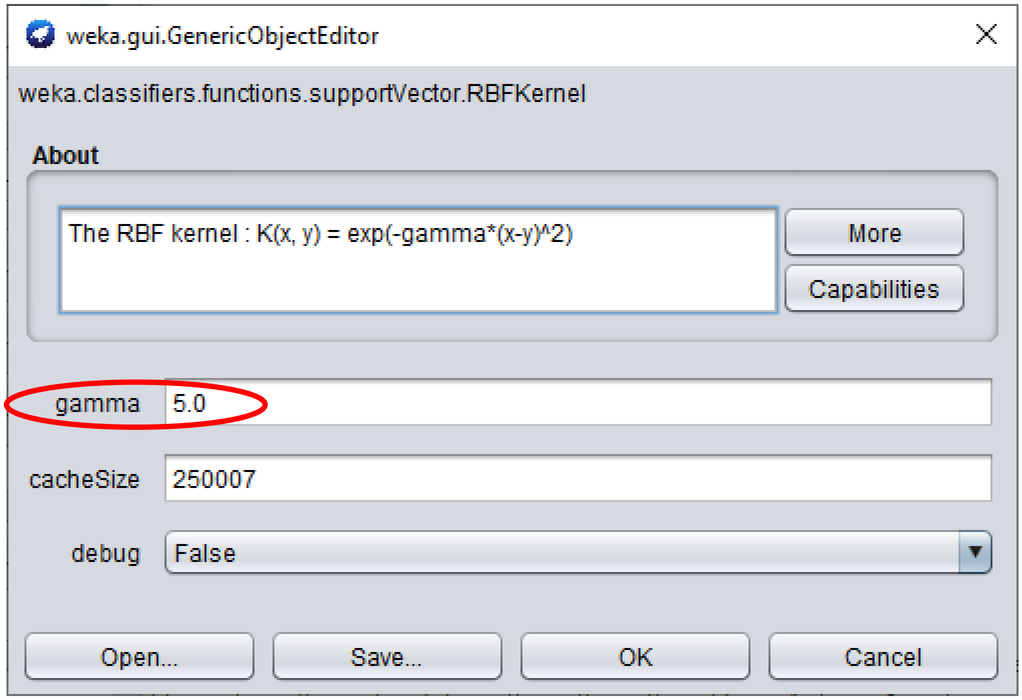


Figure 28 RBF kernel configuration.

- After trying different parameters and run the model, you can save the model with the highest overall accuracy and outputs to a text file. The SVM output is shown in Figure 29.

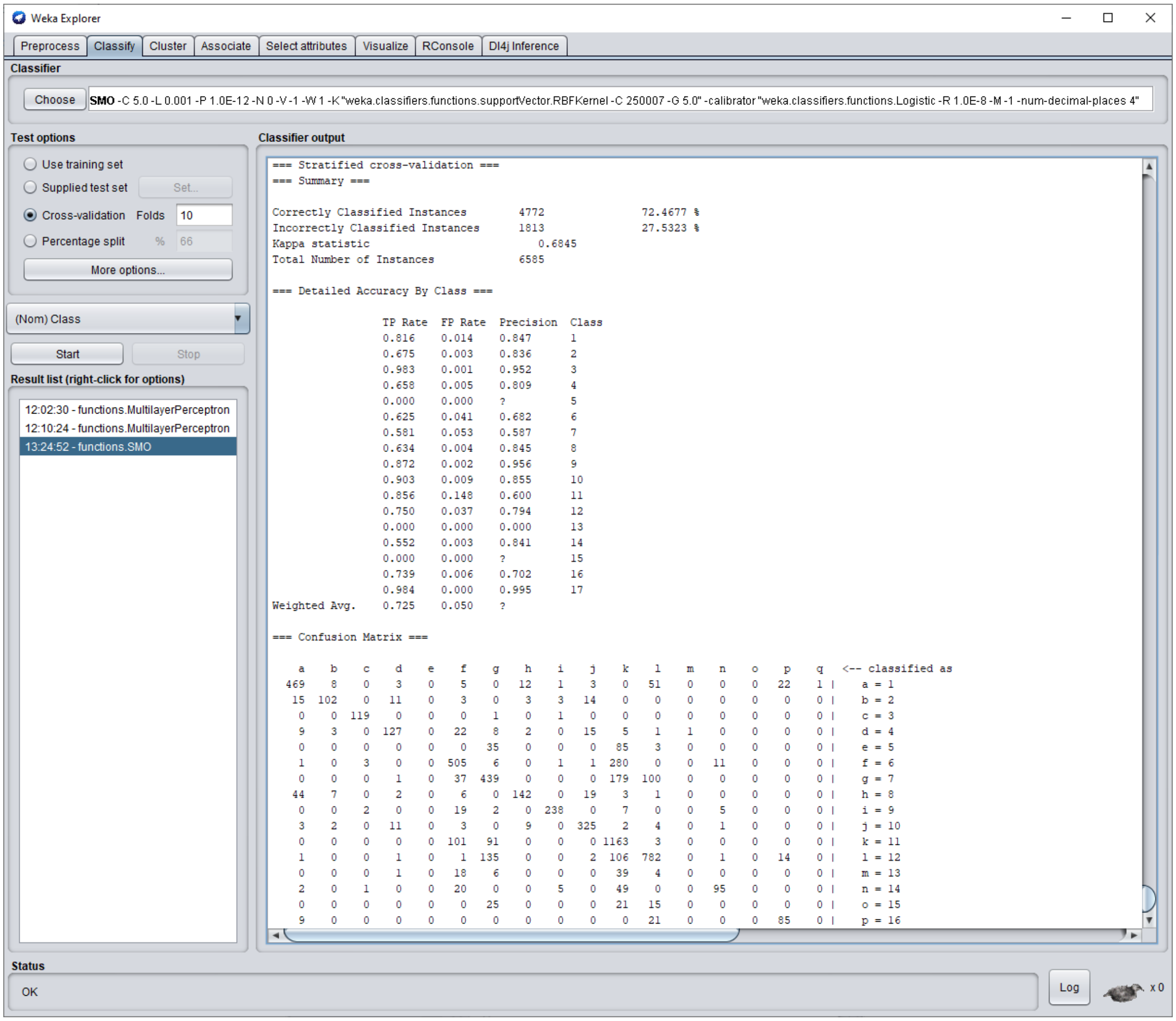


Figure 29 Output of SVM classifier.

- Running kNN: Click the "Choose" button and select "IBk" under the "lazy" group. Click on the name of the algorithm to review the algorithm configuration. kNN works by storing the entire training dataset and querying it to locate the *k* most similar training patterns when making a prediction. The size of the neighborhood is controlled by the *k* parameter; thus *k* is an important parameter. Weka can automatically discover a good value for k using cross validation inside the algorithm by setting the crossValidate parameter to True. Another important parameter is the distance measure used. This is configured in the nearestNeighbourSearchAlgorithm which controls the way in which the training data is stored and searched. The default is a LinearNNSearch. Clicking the name of this search algorithm will provide another configuration window where you can choose a distanceFunction parameter. By default, Euclidean distance is used to calculate the distance between instances, which is good for numerical data with the same scale. Manhattan distance is good to use if your attributes differ in measures or type. It is a good idea to try a suite of different *k* values and distance measures and see what works best. Here we set *k* to 5 and selected Linear NN search and Euclidean distance (Figure 30).

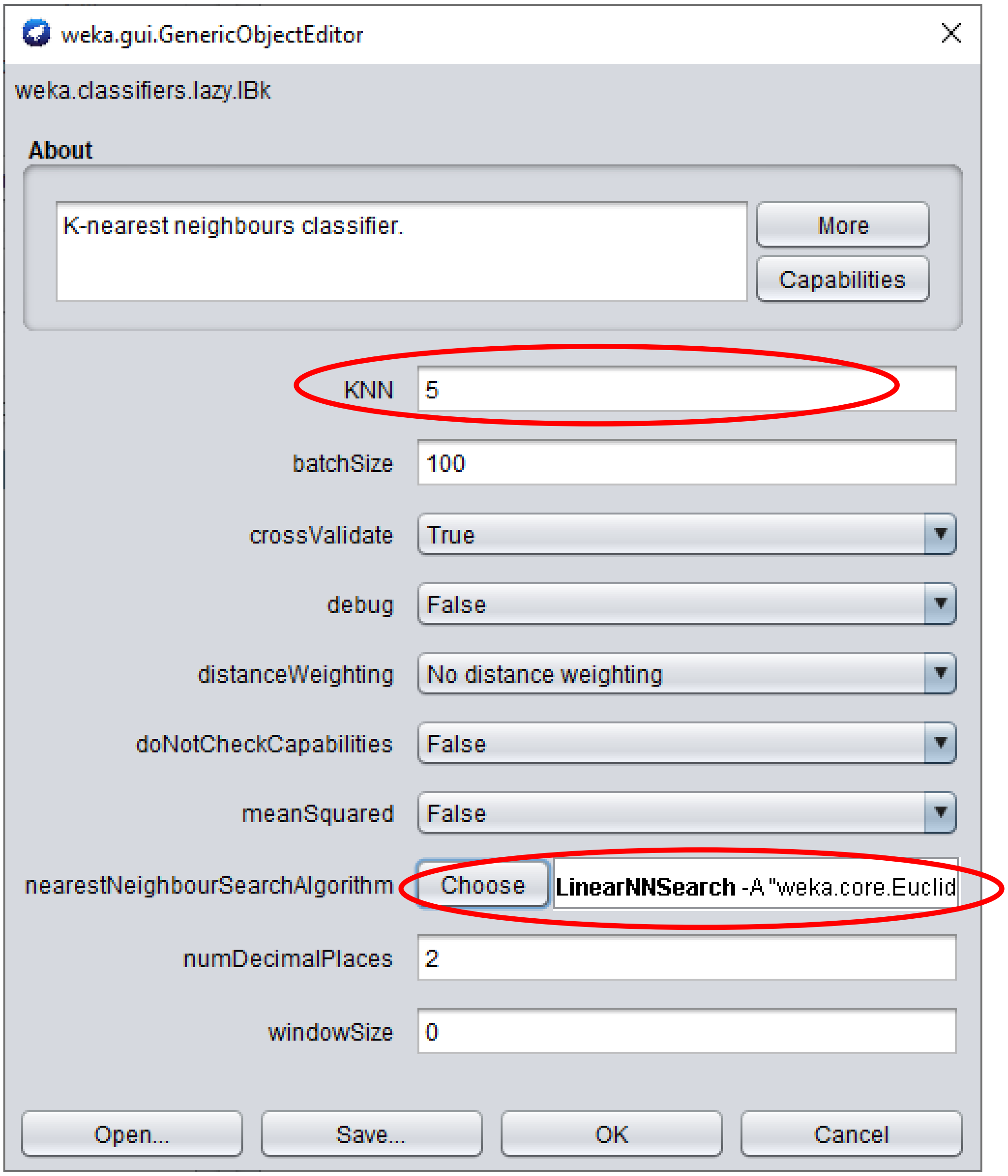


Figure 30 kNN configuration in WEKA.

- Running RF: Click the "Choose" button and select "RandomForest" under the "trees" group.
- Click on the name of the algorithm to review the algorithm configuration. Two parameters are important in RF. numIterations -- The number of trees in the random forest. numFeatures -- Sets the number of randomly chosen attributes. If 0, int(log_2(#predictors) + 1) is used. We can test different parameters. RF also can identify variable of importance by setting computeAttributeImportance to true. The interface of RF is displayed in Figure 31. Similarly, the outputs can be saved to a text file.

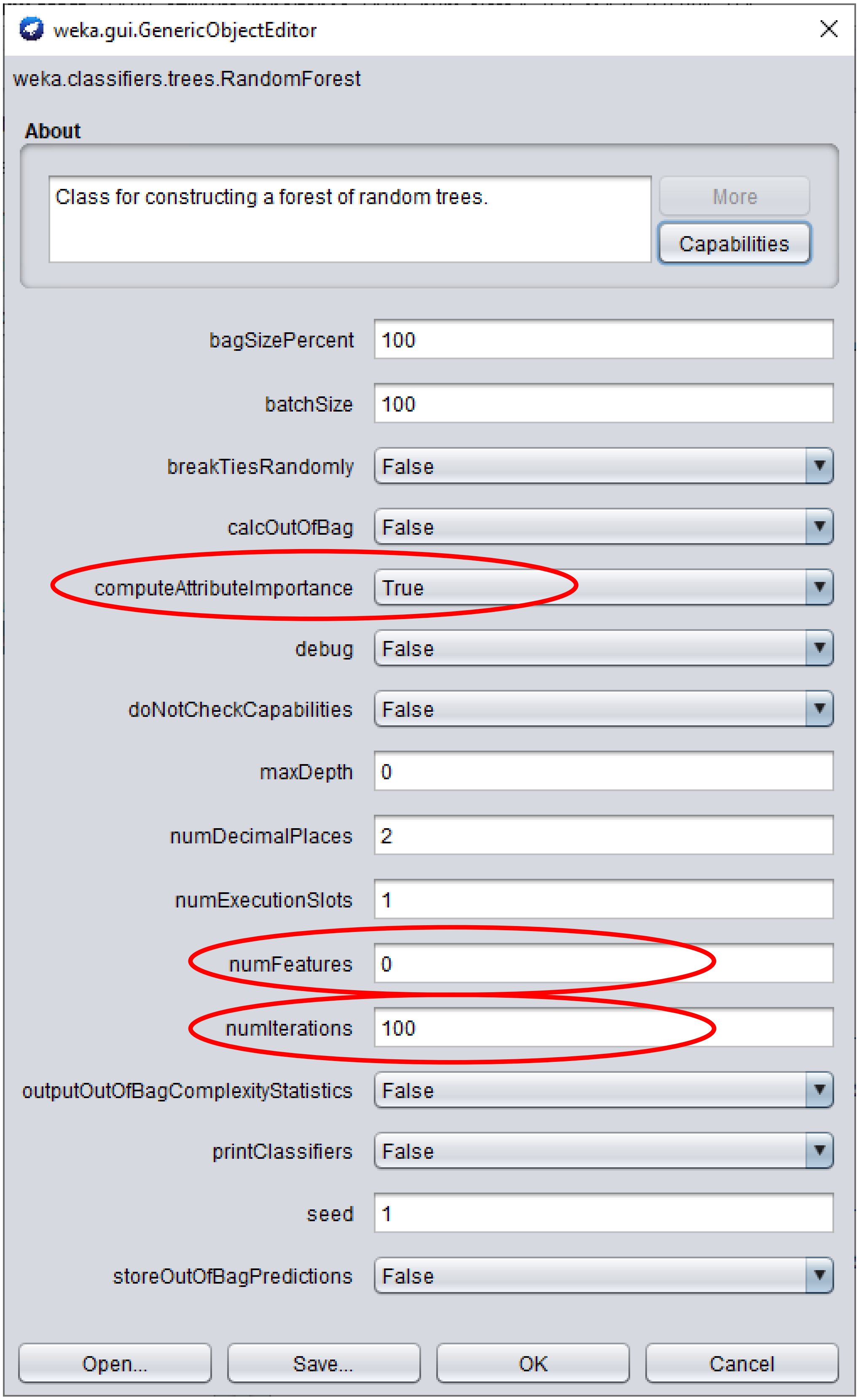


Figure 31 RF configuration in WEKA.

- You can also test other classifiers in WEKA.
- Now we need to generate predictions for all image objects based on models developed from reference samples. A similar data processing steps in ArcGIS and WEKA are needed as below.
    - In ArcGIS, in the segmentation file exported from eCognition and added DEM and NDVI features, add a new field Class as integer type, and set all values to 1, then export the attribute table to a csv file. Note: only all numerical attributes and the Class fields are exported. These features and attribute names must be same as the training data used in WEKA. A screenshot of this ARFF dataset is shown in Figure 32.
    - Similarly, load in the csv file in WEKA and save it as ARFF, and revise the Class attribute as nominal and has values same as the training ARFF. All attribute types and name must be consistent with the training ARFF.
    - Load in the training ARFF file, and click Classify tab, similarly set up the classifier, in Test options, select Supplied test set, click Set, and load in the ARFF to be predicted (Figures 33, 34).
    - Save the output in the text box of Classifier output as a text file (*.txt).

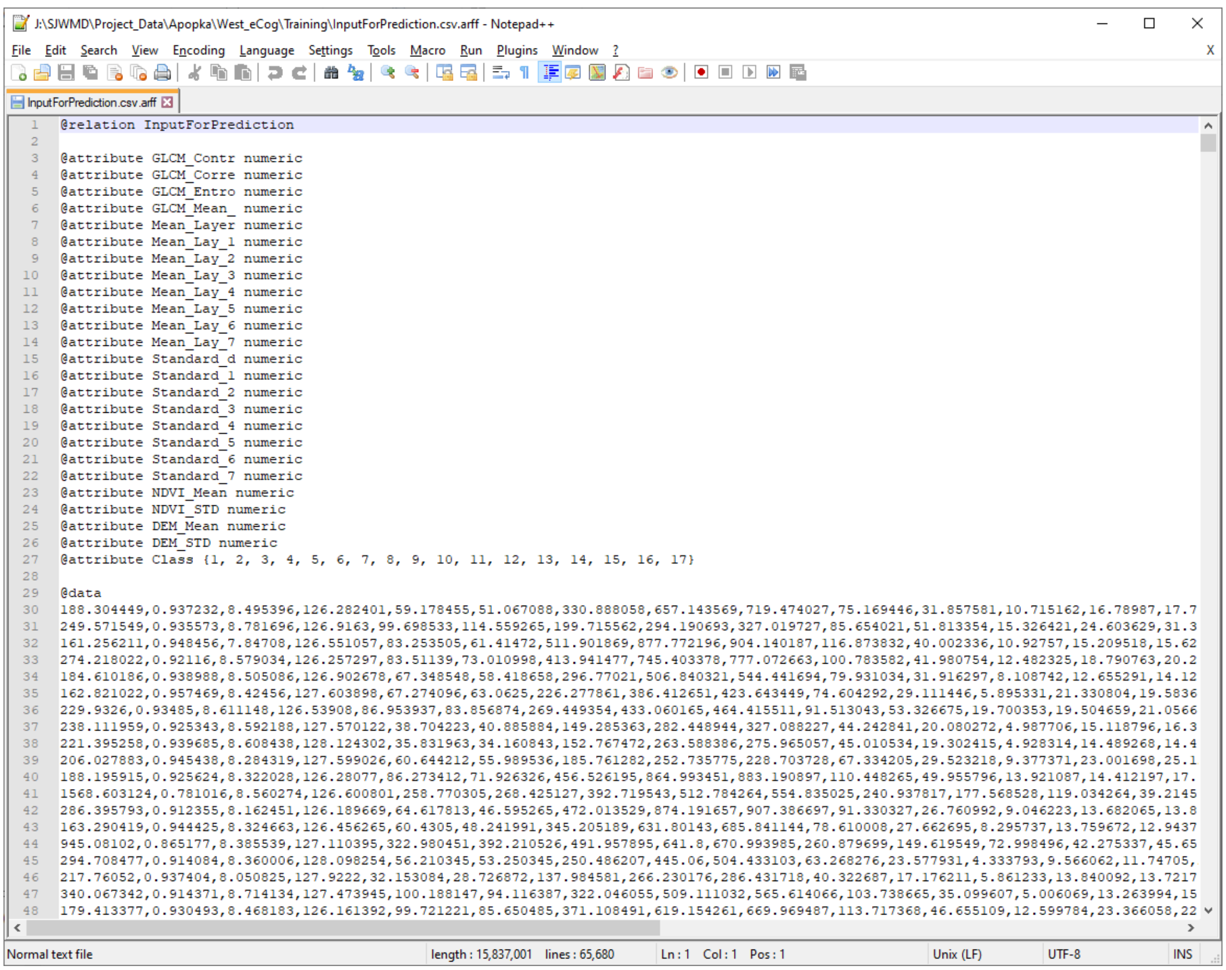


Figure 32 Input dataset in ARFF format for prediction. A total of 65650 records/instances/objects to be predicted.

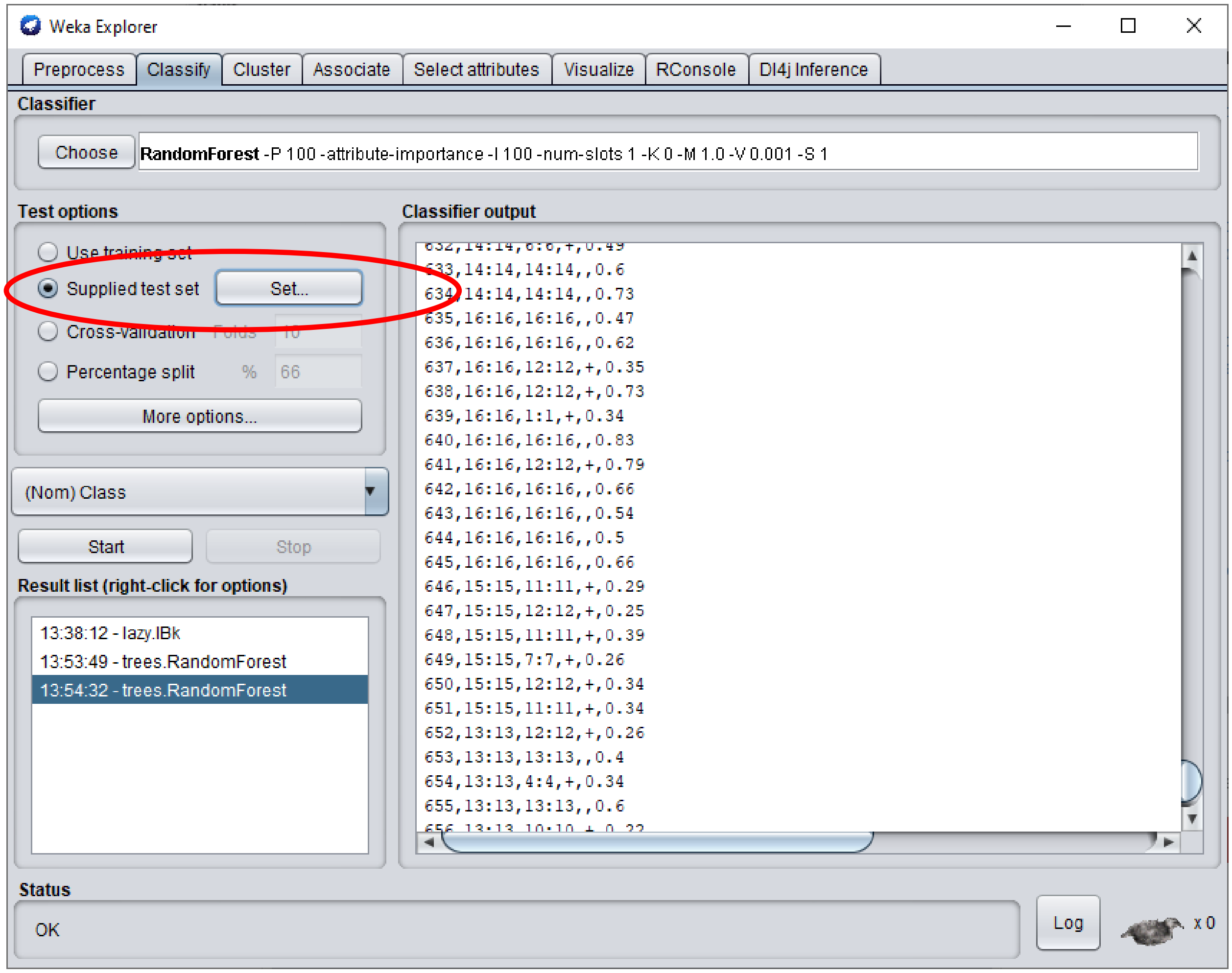


Figure 33 Setup prediction dataset.

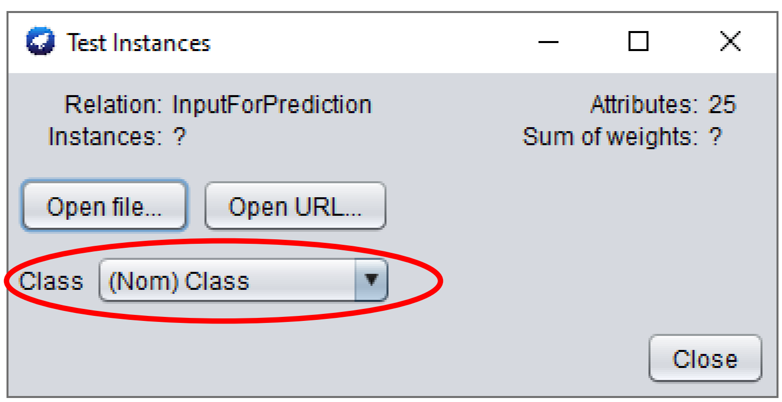


Figure 34 Ensure Class is set to Class (attribute to be predicted).

- Open the text file, revise and drop the useless information at the beginning and the end, and only keep the prediction part, as screenshot in Figure 35.

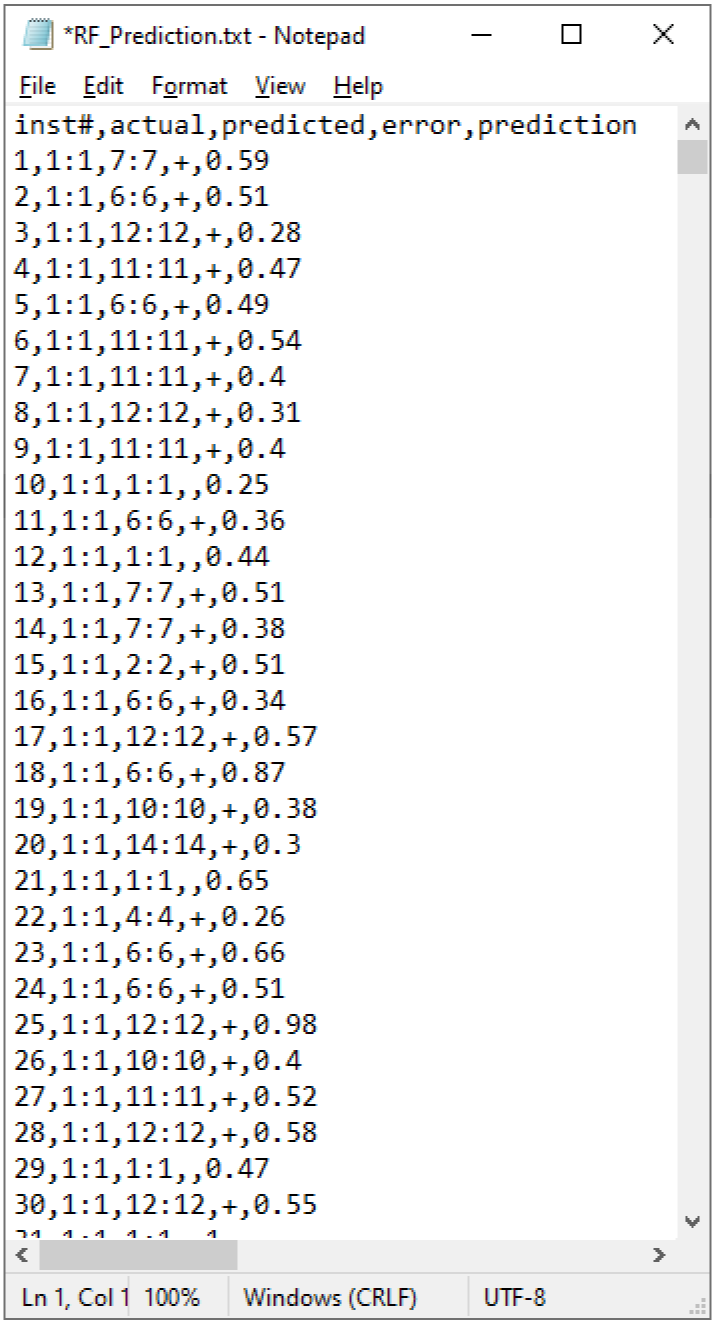
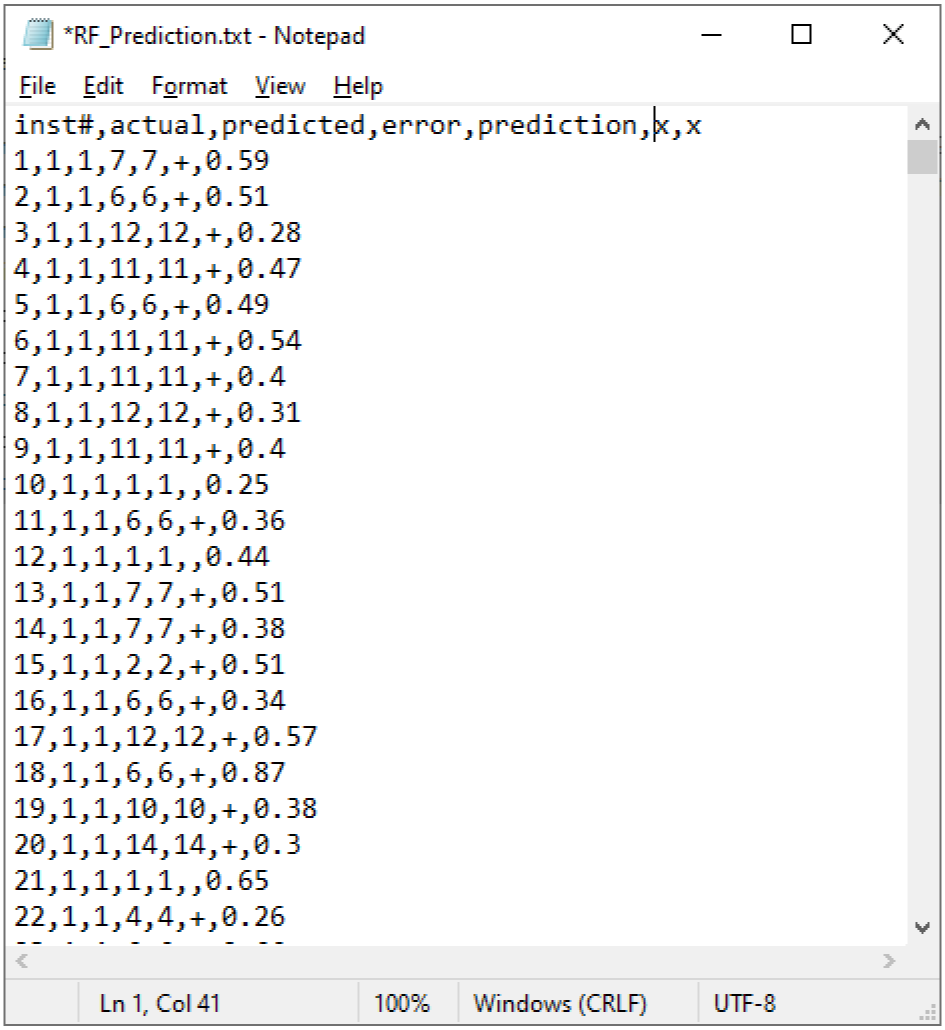


Figure 35 Output of the revised prediction text file.

- Revise this text file, replace “:” with “,”, and add “x, x” to the beginning of the file, as shown in Figure 36. This is to revise a format ArcGIS can read and join with the segmentation file to get the class prediction value.




```
inst#,actual,predicted,error,prediction,x,x
1,1,1,7,7,+,0.59
2,1,1,6,6,+,0.51
3,1,1,12,12,+,0.28
4,1,1,11,11,+,0.47
5,1,1,6,6,+,0.49
6,1,1,11,11,+,0.54
7,1,1,11,11,+,0.4
8,1,1,12,12,+,0.31
9,1,1,11,11,+,0.4
10,1,1,1,1,,0.25
11,1,1,6,6,+,0.36
12,1,1,1,1,,0.44
13,1,1,7,7,+,0.51
14,1,1,7,7,+,0.38
15,1,1,2,2,+,0.51
16,1,1,6,6,+,0.34
17,1,1,12,12,+,0.57
18,1,1,6,6,+,0.87
19,1,1,10,10,+,0.38
20,1,1,14,14,+,0.3
21,1,1,1,1,,0.65
22,1,1,4,4,+,0.26
```

Figure 36 Revised RF prediction text file to be read in ArcGIS. First column Inst# can be used to join in ArcGIS, Column prediction is the predicted class.

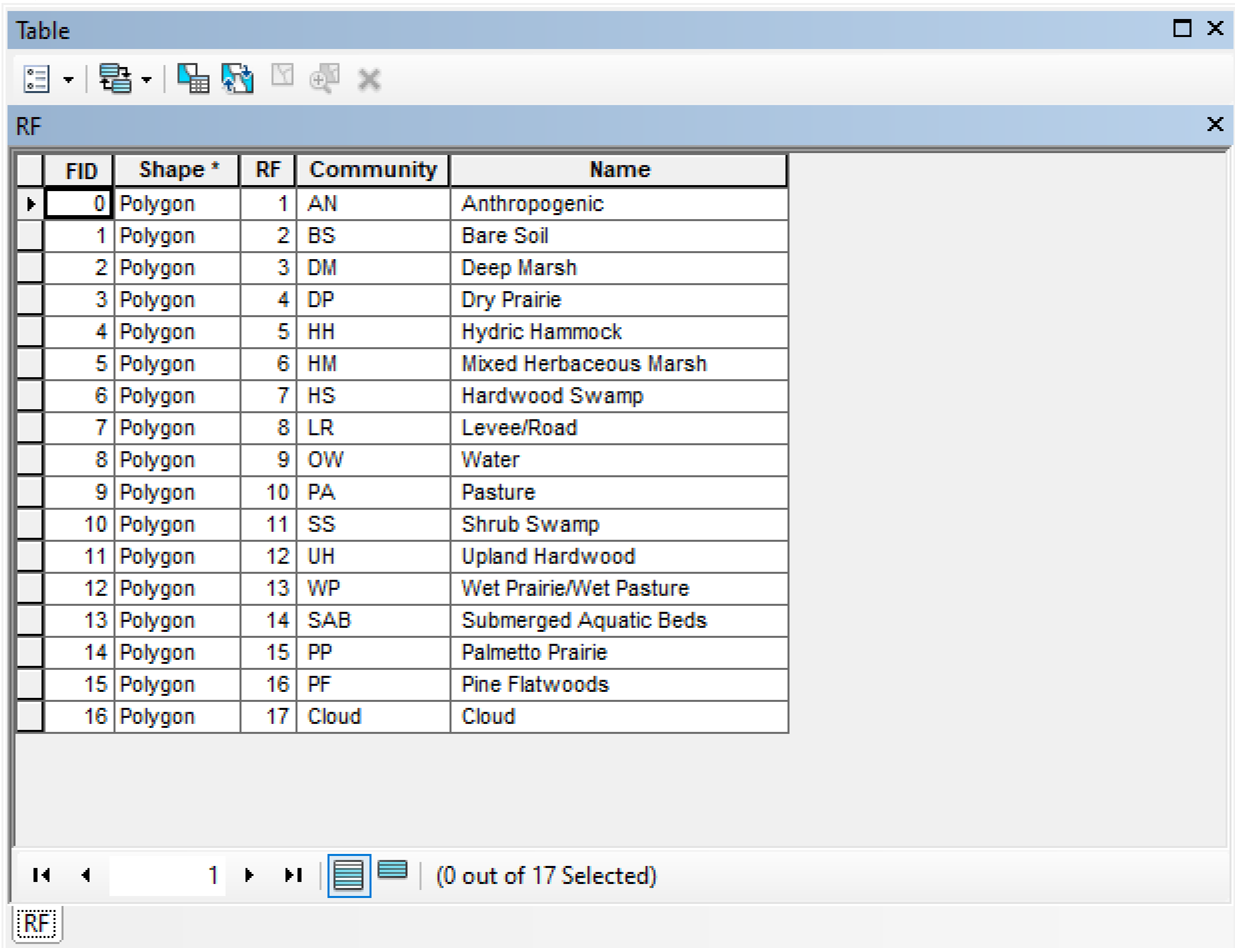
- Following similar procedure to generate prediction files from SVM, ANN, kNN or any classifier you selected in WEKA, and revise the text file readable in ArcGIS.

Part VI Final Mapping

Now machine learning training and classification have completed. The prediction results will be imported into ArcGIS. Steps are below.

- In ArcGIS, load in the segmentation file, and uncheck all fields except FID, and then export this segmentation shapefile to be used for mapping (e.g., mapping.shp). In this way, the large file size with many attributes can be largely reduced and easier to handle.
- Load in this mapping.shp, add a new integer field named as ID, and calculate this new filed as FID+1, and add integer fields named RF, SVM, ANN, and kNN.
- Load in the prediction text (*.txt, Figure 36), then join the mapping.shp with this prediction text based on ID, and Inst#, and then get the value of RF field from the prediction column. Similarly, we can get SVM, ANN, and kNN prediction, respectively.

- Add an integer Uncertainty field, select if RF=SVM, and SVM=ANN, then assign value to 1 (full agreement), and select RF<>SVM, RF<>ANN, and SVM<> ANN, then assign value to 3 (No agreement), and then change the default value 0 to 2 (partial agreement).
- Dissolve the RF, SVM, ANN, and Uncertainty fields to get the final mapping shapefile.
- Add name: load in the name text file (Figure 2), and join this text file with the final mapping shapefile to get the community abbreviations and full name to the final mapping files. An example of the attribute table of the final shapefile from the RF classification is displayed in Figure 37.



FID	Shape *	RF	Community	Name
0	Polygon	1	AN	Anthropogenic
1	Polygon	2	BS	Bare Soil
2	Polygon	3	DM	Deep Marsh
3	Polygon	4	DP	Dry Prairie
4	Polygon	5	HH	Hydric Hammock
5	Polygon	6	HM	Mixed Herbaceous Marsh
6	Polygon	7	HS	Hardwood Swamp
7	Polygon	8	LR	Levee/Road
8	Polygon	9	OW	Water
9	Polygon	10	PA	Pasture
10	Polygon	11	SS	Shrub Swamp
11	Polygon	12	UH	Upland Hardwood
12	Polygon	13	WP	Wet Prairie/Wet Pasture
13	Polygon	14	SAB	Submerged Aquatic Beds
14	Polygon	15	PP	Palmetto Prairie
15	Polygon	16	PF	Pine Flatwoods
16	Polygon	17	Cloud	Cloud

Figure 37 The attribute table of the final RF classification shapefile in ArcGIS.