Supervised kNN classifier (kNN)

Updated: 22.05.2017.

About

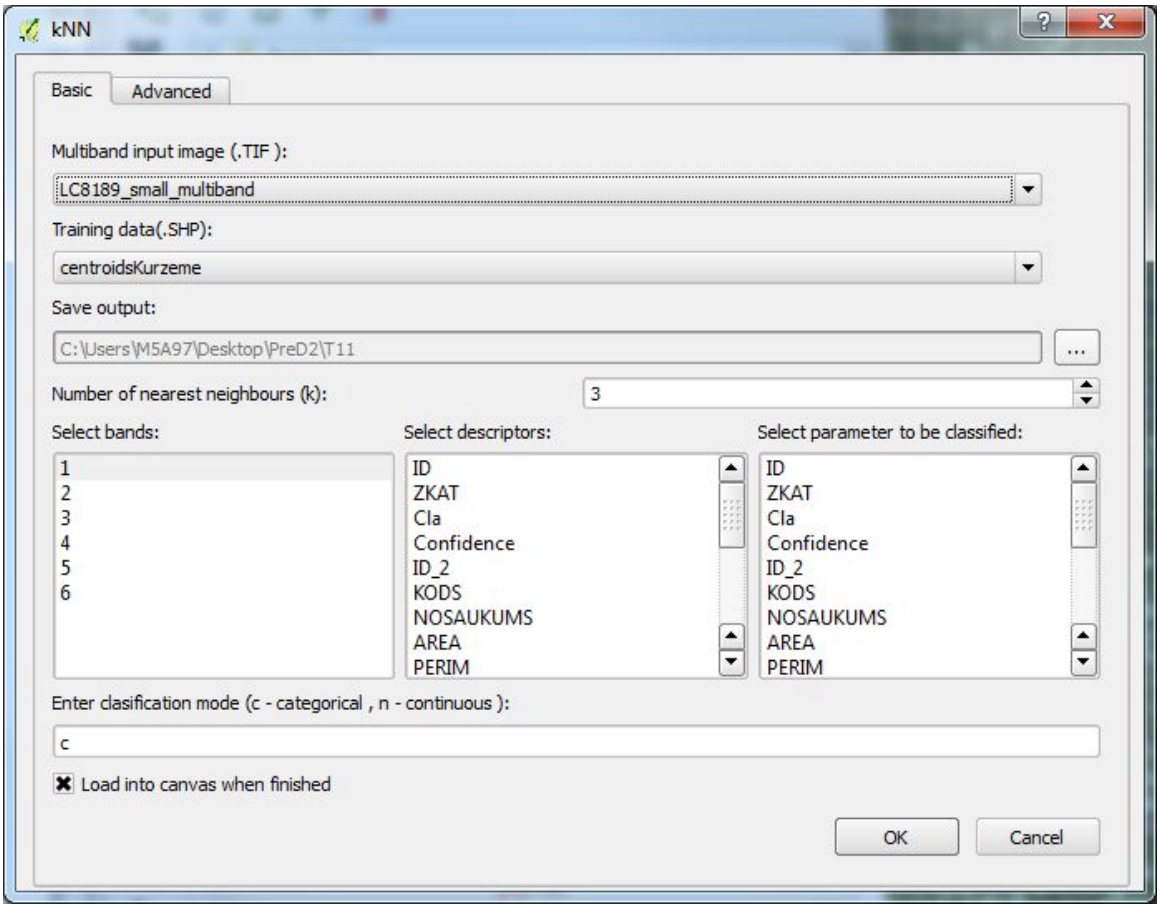
Plugin performs .TIF image pixel classification using kNN classification method.
Classification process is supervised by user provided .SHP file.
Plugin functions and working principles are implemented based on [1] paper.

LICENSE:

How to install from file?

1. Copy plugin folder to directory ~\.*qgis2\python\plugins*, where ~ is home directory. Typically for Windows: *C:\Users\UserName\.qgis2\python\plugins*, for Unix-like */home/UserName/.qgis/python/plugins*.
   Under UNIX like systems folder qgis2 is hidden, because of how system handles folders starting with dot symbol "*.qgis2*". To show hidden files and folders use *Ctrl+H* shortcut if using file browser like *Nautilus* or terminal command *ls -a* to view hidden file and folder entries.
2. Open QGIS → Plugins → Manage and install plugins → all. Find the Plugin by typing "kNN" in Search box. Add plugin by checking the tick box.

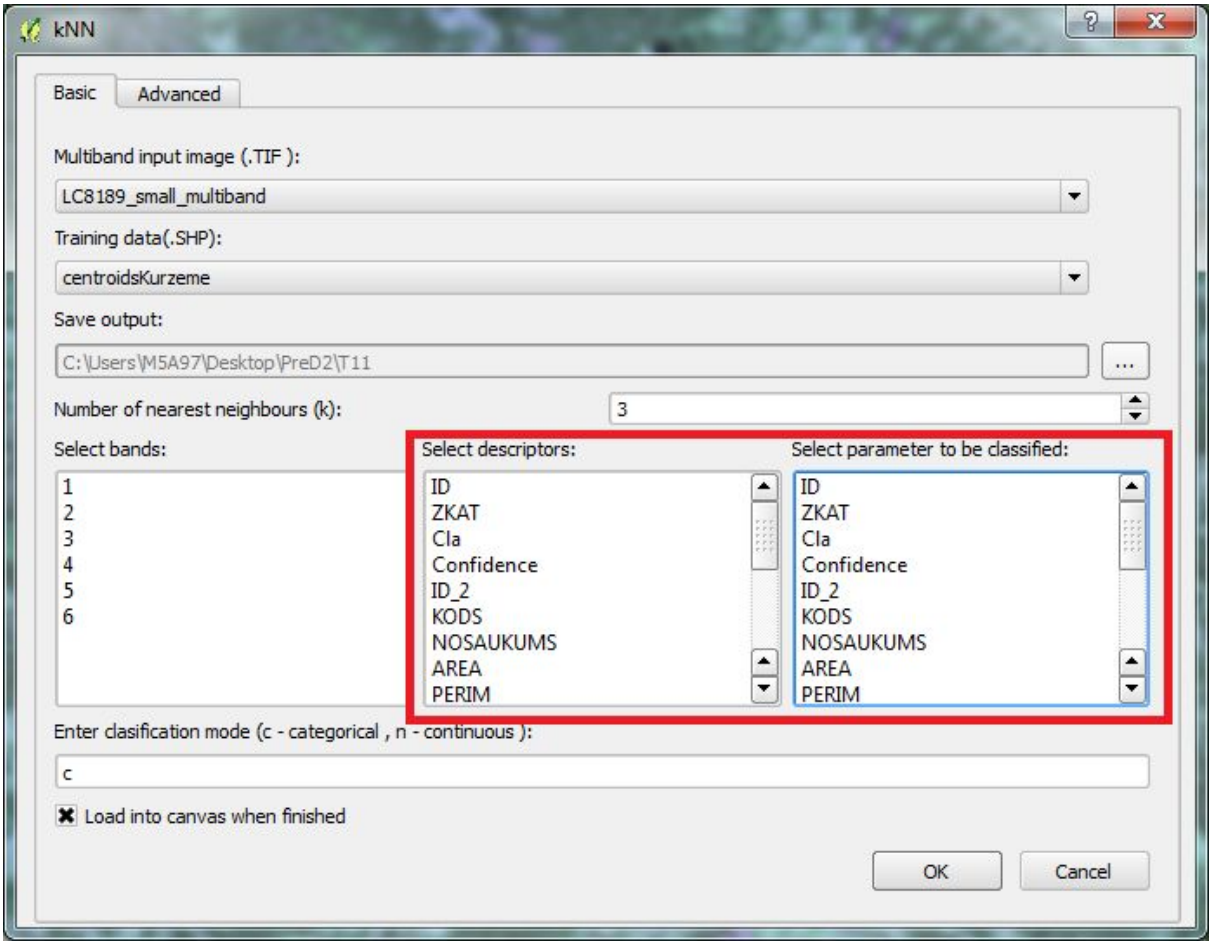
If plugin is not showing up be sure to check settings and set option *Show also experimental plugins*. Option can be found QGIS → Plugins → Manage and install plugins → Settings



Input (Basic)

In the order for plugin to operate user must provide it with the necessary input.

1. Multiband input image: image (.TIF extension) which needs to be processed. In the current example “LC8189_small_multiband” is provided. Example image consists of 6 raster bands.
2. Training data: shape file (.SHP extension), contains descriptors and parameters to be classified. In current example “centroidsKurzeme” shape file is provided. User must calculate descriptors and prepare parameters before using the plugin (for example, descriptor values for polygon .SHP files can be calculated using Zonal Statistics plugin). Shape file header contents will fill in descriptor and parameter list with the same content. At this point user is responsible of the correct selection of the descriptors and parameters to correctly start the classification process.



Contents of .SHP file are shown in both descriptor and parameter fields.

3. Save output: Plugin must be provided with the save directory. Plugin generates file names automatically and saves in the provided directory.
   Generated filename: SC_ + input image name + parameter name +_ k value + mode
   Accuracy assessment TXT file will be saved in the same directory.
   Accuracy assessment filename: SC_ + input image name + Acc.
4. Number of the nearest neighbors: number of the nearest sample descriptor vectors according to the distance.
5. Select bands: list of the available bands for current image file. Number of the elected bands must match number of the selected descriptors.
6. Select descriptors : list of the shape file contents. User must provide descriptor relevant fields only.
7. Select parameter to be classified: user must provide classifiable parameter. Multiple parameters also can be classified. Classification of multiple parameters is recommended as it saves time.
8. Classification mode: depending on classifiable parameter user can specify classification mode. For categorical parameter - c, for continuous parameter - n . Entered modes are separated by "," symbol for each selected parameter. Categorical variables contain a finite number of categories or distinct groups. For example,

categorical predictors include tree species and land cover type. Continuous variables are numeric variables that have an infinite number of values between any two values. For example, stand volume, stand age, stand height etc.

Following example shows how classification result minimum and maximum values look using classification mode - n.

Following example shows how classification result minimum and maximum values look using classification mode - c.

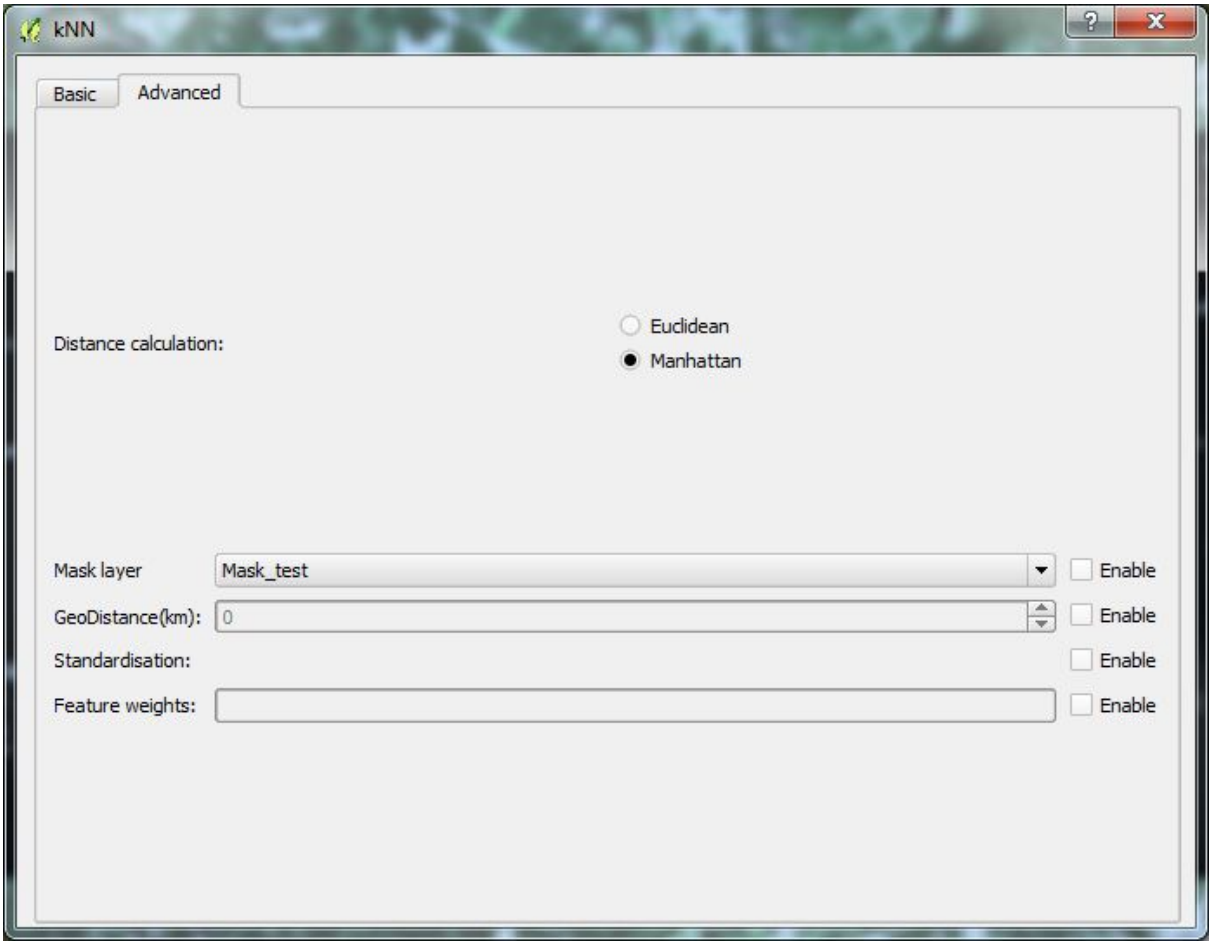
9. Tick box : When enabled (ticked), it loads results in the current QGIS workspace.

User must manage files in the selected save directory. When classification is done in the selected directory following files with their appropriate file names will be saved:

- Generated file name (.TIF): SC_ + input image name + parameter name +_ k value + mode.
- Accuracy assessment file name (.TXT): SC_ + input image name + Acc.

If user will not rename saved files or change save directory and will repeat classification process, only the latest version of result will be saved. In this case result gets overwritten with latest classification data.

If multiple entries from list have to be selected (for example, bands or descriptors) Ctrl key is used. Any other methods of selection (MouseClickDrag, Shift etc.) may cause input data errors.



Input (Advanced)

1. Distance calculation options affect speed and results of the calculation. Each of the options uses different formula for calculations.

   Euclidean distance. Significantly slower than Manhattan distance calculation because of square root calculations. Euclidean distance is used to calculate distance in the feature space. In this case, general formula for calculation of the distance in *n* dimensional feature space is :

   $$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2}$$

   where :

   p_n - *classifiable pixel descriptor value vector element.*

   q_n - *sample data descriptor value vector element.*

   Manhattan(faster). Faster than Euclidean method, but does not yield exactly the same result.

   $$d_1 = (p,q) = \|p - q\|_1 = \sum_{i=1}^{n} |p_i - q_i|$$

   where :

p_n - *classifiable pixel descriptor value vector element.*

q_n - *sample data descriptor value vector element.*

In case of categorical variable most common category of k nearest categories from classifiable pixel is assigned.

In case of numerical variable all of the nearest neighbor values are taken in consideration, but each of them has weighed proportionally to the distance from classifiable pixel. In this case following formulas are used :

$$w_{(p_i)p} = \frac{1}{d^t_{(p_i)p}} \Big/ \sum_{j=1}^{k} \frac{1}{d^t_{(p_i)p}}$$

where:

$w(pi)p$ - vector of weighting values,

$d^t_{(p_i)p}$ - feature space distance from classifiable (p) and sample data(p_i).

$$\widehat{m}_p = \sum_{i=1}^{k} w_{(p_i)p}\, m_{(p_i)},$$

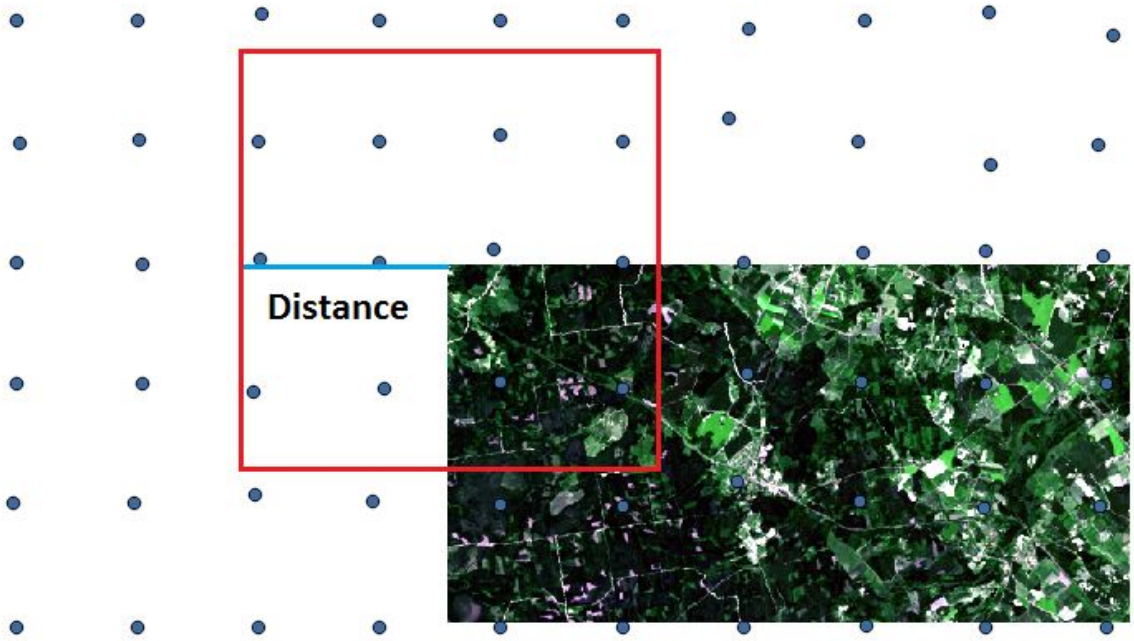
where:

$\widehat{m_p}$ - numerical value of the parameter to be classified for the classifiable pixel,

$m_{(p_i)}$ - value of the parameter to be classified for nearest neighbor pixel p_i.

2. Mask gives user option to specify which of the pixels will be classified. Mask size needs to match image dimensions. Mask value 1 - classify pixel, 0 - skip pixel.
3. Geographical distance gives user an option to select sample data in the desired distance from classifiable pixel. This option limits sample data and reduces the influence of the samples from very far location. Distance is measured in kilometers [km]. By enabling this option, user must verify that training data has columns named x and y containing coordinates of the training data points. If coordinate data is missing but feature is enabled plugin will not work properly.

   If feature is enabled but there are not enough sample data the plugin will use full range of sample data. Plugin will prompt informative message in case of insufficient sample data.



The following image shows example of how geo distance value determines which points are included in sample data. In the current example, blue line shows geo distance. Red box shows area from which sample data will be gathered. This example shows the area of the sample data for the first pixel. In this example sample data would consist of 11 entries.

If k value in this case would be < 12 plugin will successfully be able to classify pixel with given sample date.

If k value would be > 12 then classification would not be able to be finished. In this case message about insufficient sample data would pop and plugin would switch and use full range of sample data without restrictions of geo distance.

4. Standardization

$$z = \frac{x - \mu}{\sigma}$$

where:

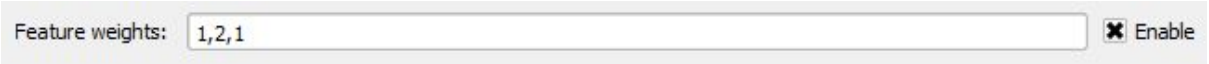
μ is the mean of the descriptor values.

σ is the standard deviation of the descriptor values

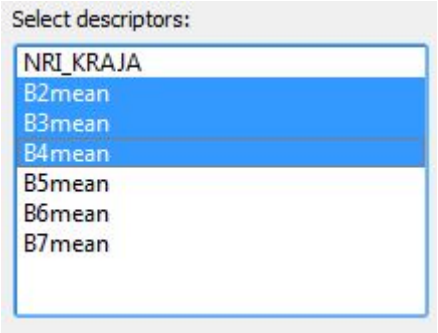
5. Feature weights function as coefficients which affect how great impact data from each descriptor has. Coefficient < 1 reduces descriptor significance coefficient > 1 increases descriptor significance. Entered weights are separated by comma. If

feature weights are used, number of entered weights must be equal to number of the selected descriptors.

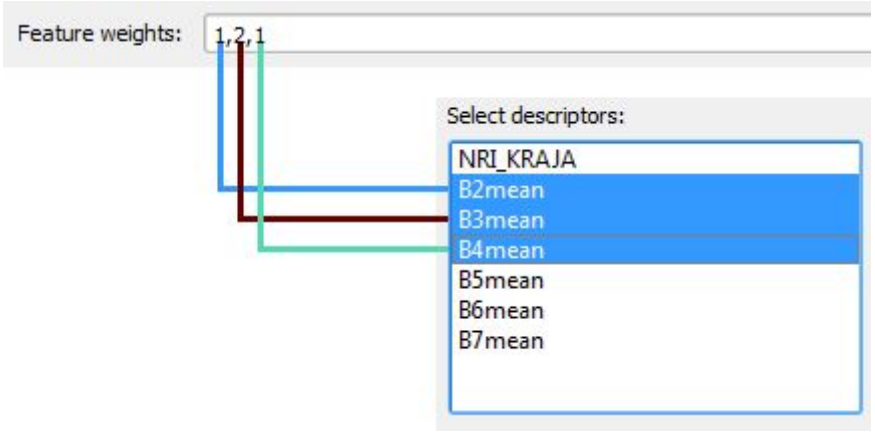
Example:



As we can see in current example there are three feature weights entered. In this case we have to also have three descriptors selected like this :



Each weight coefficient corresponds to a specific selected descriptor. In current example descriptor “B3mean” will have greater significance than “B2mean” and “B4mean”. Changing significance of descriptors changes classification result.



General distance formula

$$d_{p(pi)} = \sqrt{\sum_{j=1}^{nf} (x_{p,j} - x_{(pi),j})^2}$$

where:

$d_{p(p_i)}$ - distance from classifiable (p) and sample data(p_i)

$x_{p,j}$ - classifiable pixel (p) feature No. j value,

$x_{(p_i),j}$ - sample data(p_i) feature No. j value

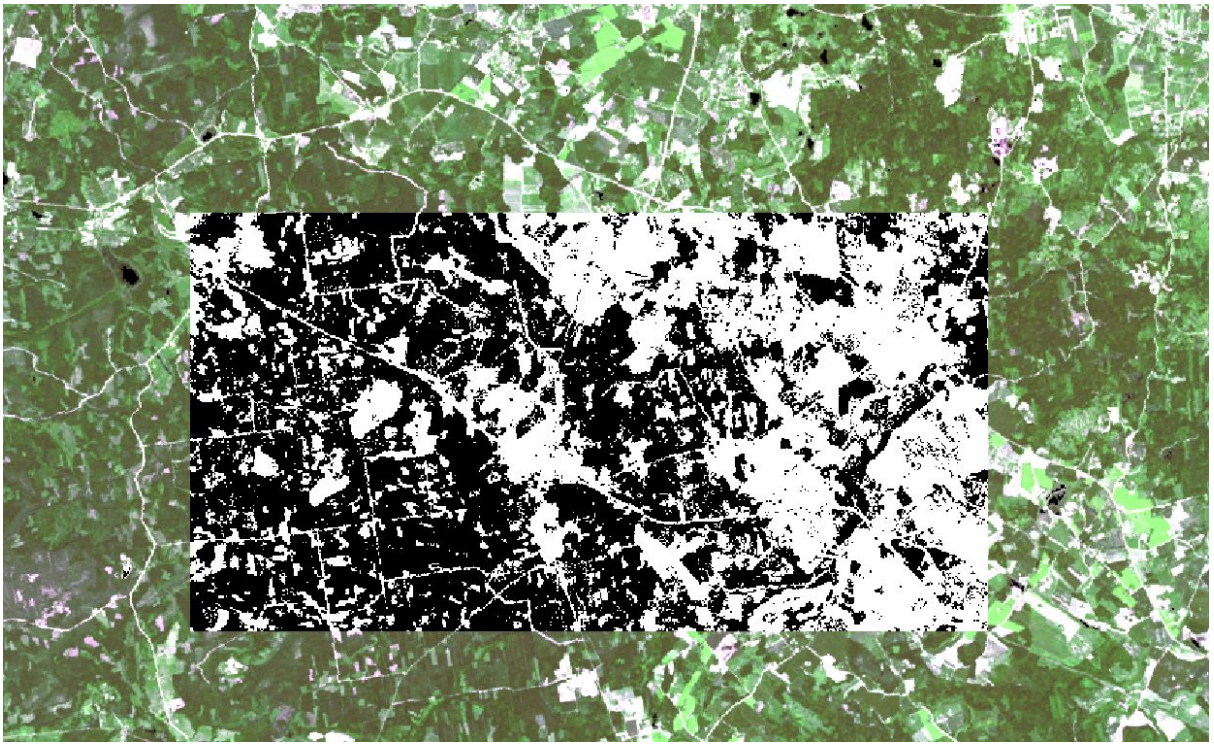
nf - number of fatures

Weighted distance formula

$$d_{p(pi)} = \sqrt{\sum_{j=1}^{nf} a_j^2 (x_{p,j} - x_{(pi),j})^2}$$

where:

a_j^2 - feature weight coefficient vector



Example classification result. Chunk of multiband map classified into two groups. In this example classification is executed in C-mode. Pixels are assigned to specific class.

This example shows result when classification is executed in N-mode. Pixels are described with continuous values which can be quantitatively compared.

Output

.TIF file/files containing classified pixels
TXT file containing accuracy assessment.

Accuracy assessment

Classifiable parameters have multiple figures which describes its accuracy:

OA - overall accuracy - all correctly classified(all types)/all classifiable(all types)
PA - producers accuracy: shows percentage how many pixels of each type are classified correctly. Correctly classified (single type) divided by the total number of test pixels of the same type according to the reference data. / correctly classified(all types)
UA - users accuracy: shows probability that specific type truly represents that type. - correctly classified (single type)/ classified(single type)

$$OA = \frac{Number\ of\ \ correctly\ \ classified\ pixels}{Number\ of\ all\ test\ pixels}$$

$$PA = \frac{correctly\ classified\ (single\ type)}{number\ of\ test\ pixels(single\ type)}$$

$$UA = \frac{correctly\ classified\ (single\ type)}{classified(single\ type)}$$

KHAT value describes how well the kNN algorithm classifies pixels if compared to random classifier.

- If KHAT < 0 : Classification is very poor. Performance is considered worse than random based classifier.
- If KHAT = 1 : All pixels classified properly
- If KHAT = 0 : Classification is random based

$$KHAT = \frac{observed\ accuracy - chance\ agreement}{1 - chance\ agreement}$$

Usually KHAT value will not reach described extremes and will be in range between 0 and 1. Example : KHAT = 0.75 which means that classification is by 75% better than one which random classifier provides.

Continuous parameter classification is described using RMSE and NRMSE value:

$$RMSE = \sqrt{\frac{\sum_{t=i}^{N} (\hat{y}_t - y_t)^2}{N}}$$

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N} (\hat{y}_t - y_t)^2}{N}}$$

$\hat{y_t}$ - predicted value (estimated by kNN) of the parameter to be classified.
y_t - value of the parameter to be classified according to the reference data.
N - number of pixels.

The RMSE of a classified pixels with respect to the valid sample data maximum and minimum.

$$NRMSE = \frac{RMSE}{(max(y) - min(y))}$$

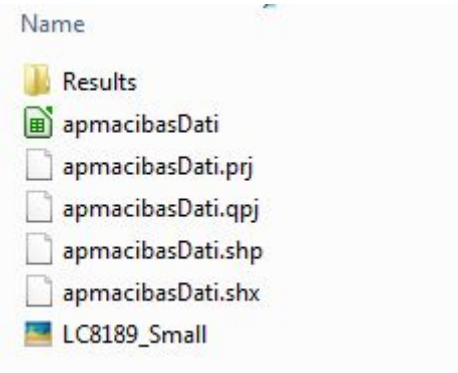
$max(y)$ - maximum value of the parameter to be classified according to the reference data,
$min(y)$ - minimum value of the parameter to be classified according to the reference data.

References

1. Hector Franco-Lopez, Alan R. Ek, Marvin E. Bauer, Estimation and mapping of forest stand density, volume, and cover type using the k-nearest neighbors method, Remote Sensing of Environment, Volume 77, Issue 3, September 2001, Pages 251-274, ISSN 0034-4257, https://doi.org/10.1016/S0034-4257(01)00209-7.
(http://www.sciencedirect.com/science/article/pii/S0034425701002097)
Keywords: Forest inventory; k-nearest neighbors; Estimation

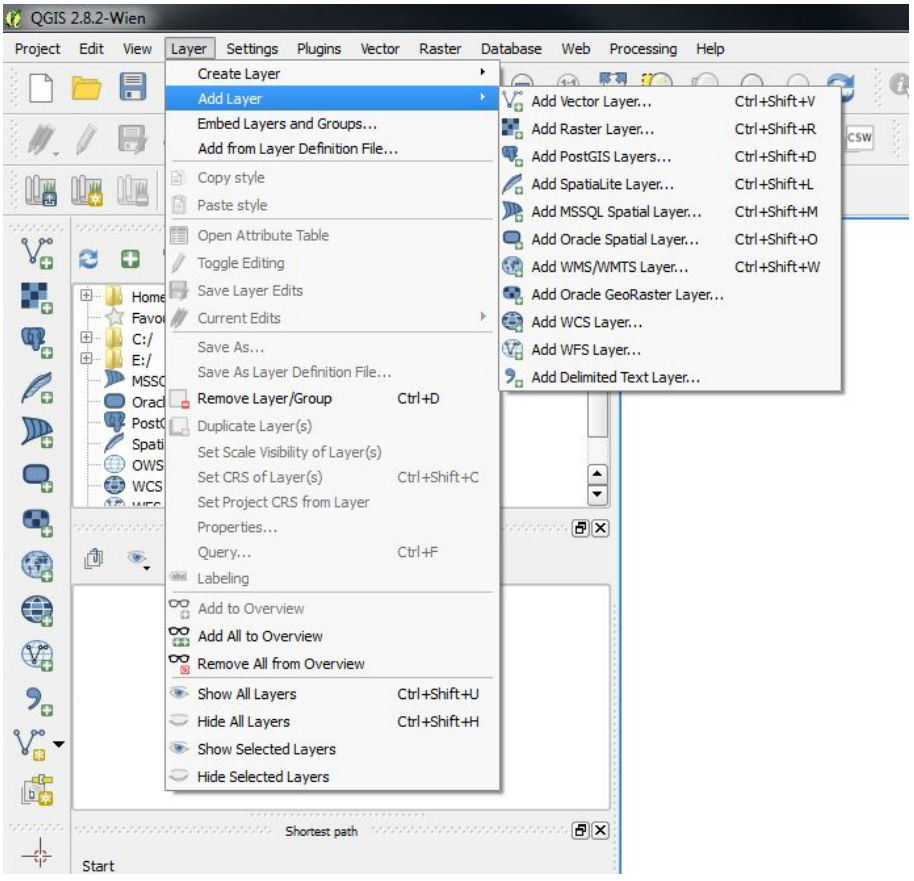
Tutorial

This segment of documentation will cover one example of classification. Plugin folder should include folder named “SAMPLE”. Contents of folder should look something like this :

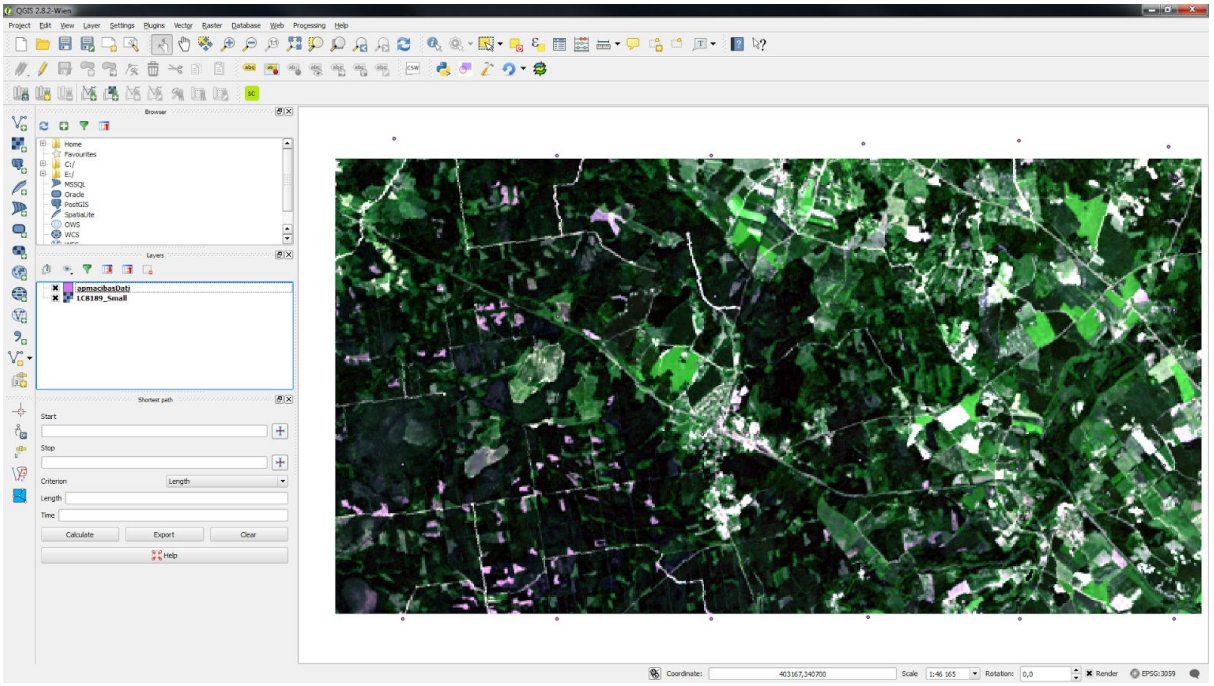


Files in this case *apmacibasDati.shp* is shape file and *LC8189_Small* is image which needs to classified. Folder named *Results* contains results of current tutorial, your results should look the same.

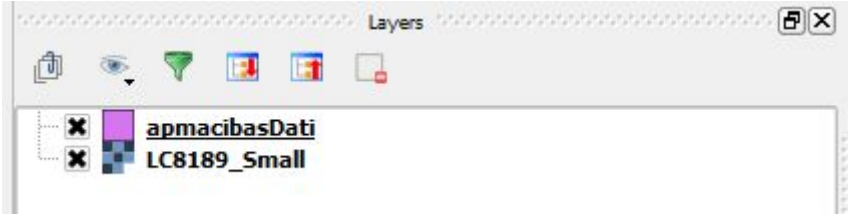
Open QGIS (in current example QGIS 2.8.2-Wien version is used)
Add *LC8189_Small.tif* file from “*SAMPLE*” folder. To add raster file (.TIF) use toolbar and navigate to *Layer → Add layer → Add Raster Layer,* or use shortcut Ctrl+Shift+R and select raster layer(.TIF file) named *LC8189_Small*.



Add *apmacibasDati.shp* file from "*SAMPLE*" folder. To add shapefile (.SHP) use toolbar and navigate to *Layer → Add layer → Add Vector Layer,* or use shortcut Ctrl+Shift+V (Source type : File) and select vector layer(.SHP file) named *apmacibasDati.shp* .



After described actions your QGIS workspace should look like this. At this point classifiable raster layer and sample data vectors should be visible. Layer window can be used to verify how many layers are present at current workspace.



Open kNN plugin *Plugins → kNN → kNN classifier*. User interface with kNN options should open.
Under dropdown menu "*Multiband input image(.TIF)*" select *LC8189_Small*
Under dropdown menu "*Training data (.SHP)*" select *apmacibasDati*
Select folder in which to save the results

Set parameters :

Basic

Number of nearest neighbors	:	3
Bands	:	*1 2 3*
Descriptors	:	*B2mean, B3mean, B4mean*
Parameter	:	*Cla*

Mode : *C*

Load into canvas when finished : *enabled*

Advanced

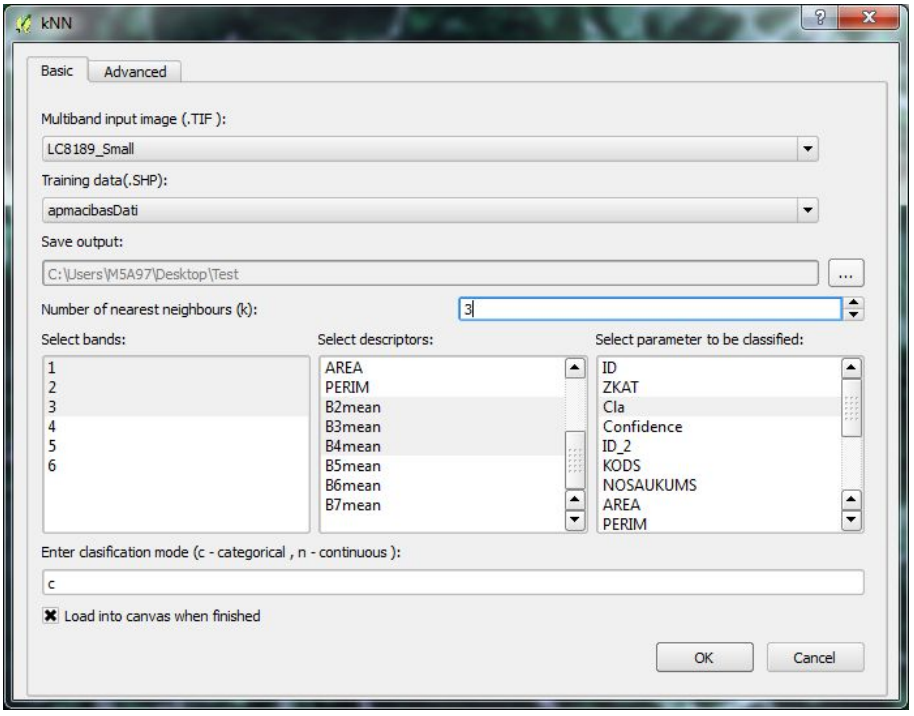
Distance : *Manhattan*

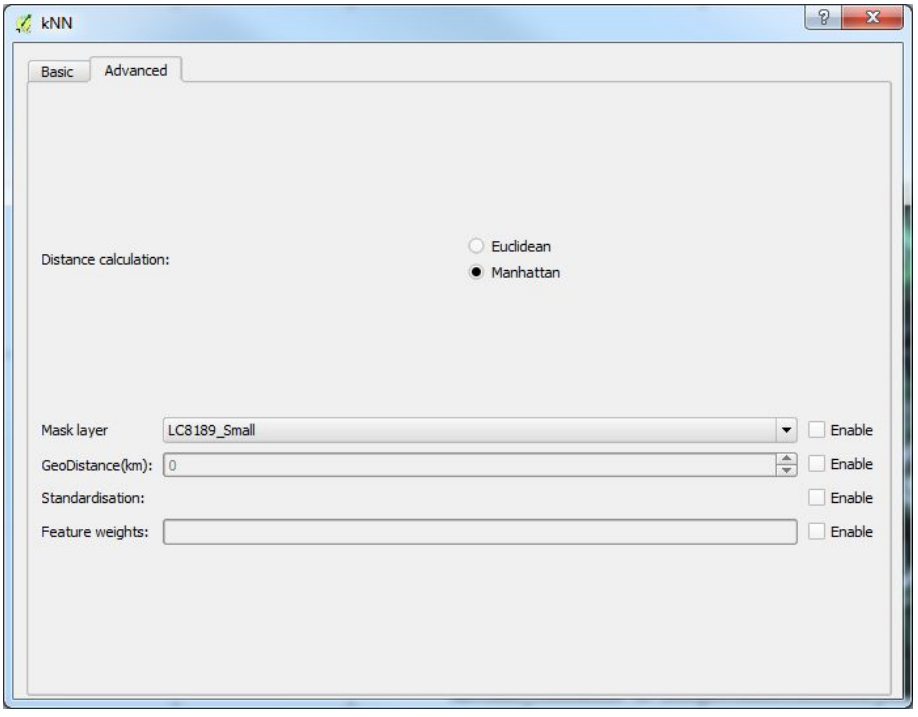
GeoDistance : *disabled*

Standardization : *disabled*

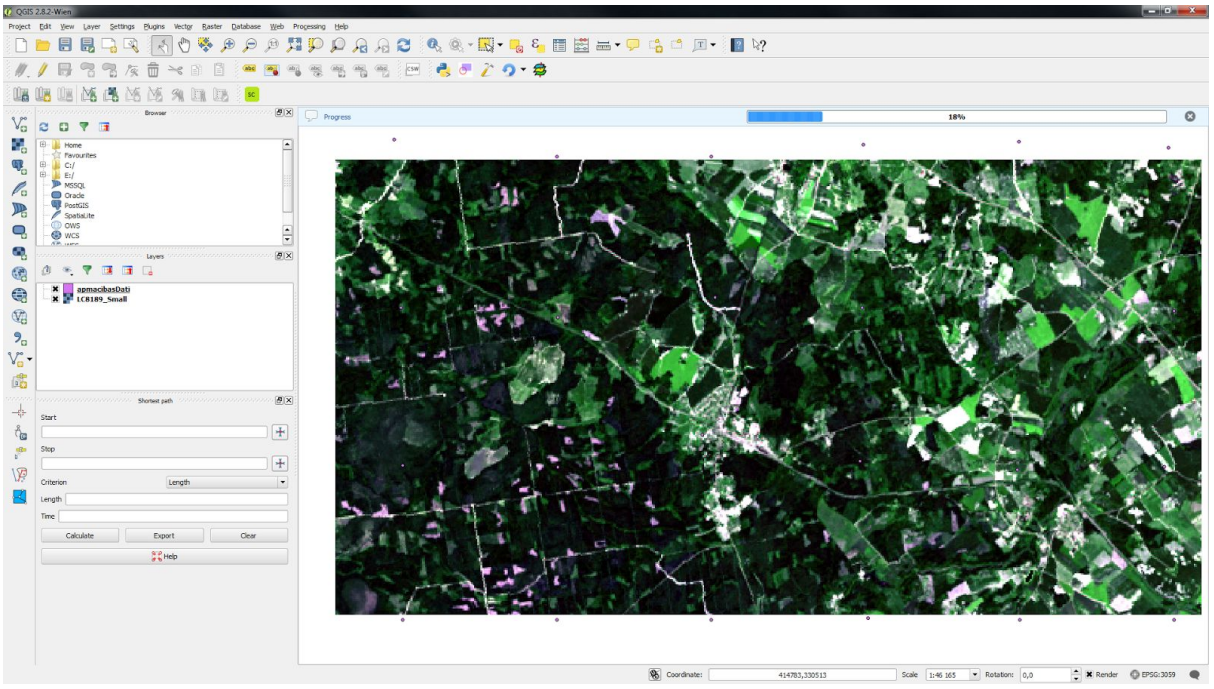
Feature weights : *disabled*

Correctly set parameter windows should look like this :

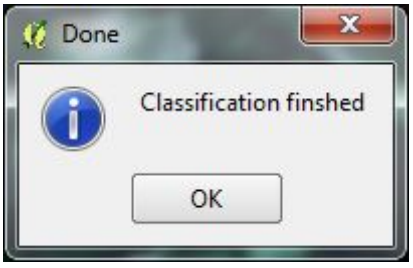




Classification will start after pressing OK button. If classification is started correctly loading progress bar should appear.

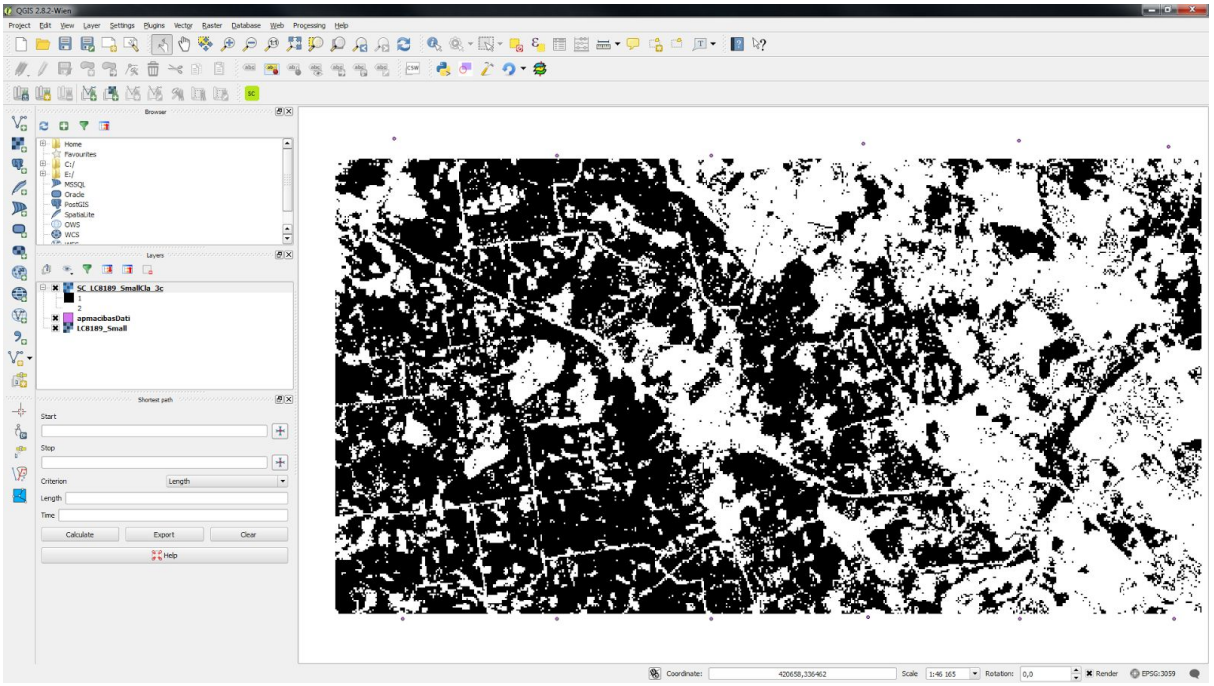


Processing time of current image should not take long time (~1 minute) . After successful classification. Message should pop.

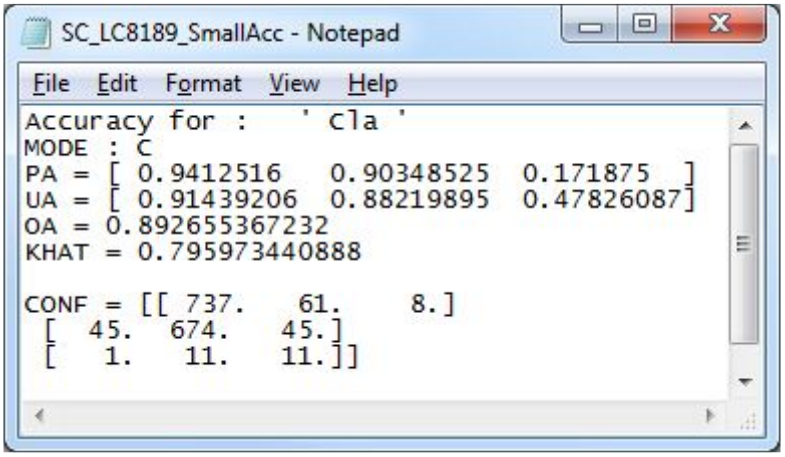


Layers window should also be populated with new layer *SC_LC8189_SmallCla_3c* .

Result of classification should also be seen:



Save folder should contain two files, raster layer and accuracy assessment . For current example accuracy assessment should look like this:



Using kNN in scripts

If user wants to use classifier via script example code is provided:

```
from supervisedClassification import supervisedClassification
sc=supervisedClassification()

imageFN="C:\\Users\\M5A97\\Desktop\\SAMPLE\\LC8189_Small.tif"
trainingFN="C:\\Users\\M5A97\\Desktop\\Misc\\centroidsKurzeme.shp"
whichBands=[1,2,3]
descriptorNames=["B2mean","B3mean","B4mean"]
k=10
variableNames=["Cla"]
outputPath="C:\\Users\\M5A97\\Desktop\\PreD2\\SaveScript"
modeList=['n']
iface= 0
FeatWght= []
Wght_En= 0
Std_En= 0
distSel=1
geoDist= 10
geoDist_En= True
mask = "C:\\Users\\M5A97\\Desktop\\SAMPLE\\LC8189_Small.tif"
mask_en = 0

rez =
sc.knn(imageFN,trainingFN,whichBands,descriptorNames,k,variableNames,outputPath,mode
List,iface,FeatWght,Wght_En,Std_En,distSel,geoDist,geoDist_En,mask,mask_en)
```

Provided code must be modified with user specific file paths and other options. Each of the parameters passed to kNN classifier must have correct data type.

imageFN	- string (Path to .TIF file)
trainingFN	- string (Path to .SHP file)
whichBands	- int array
descriptorNames	- string array
k	- integer
variableNames	- string array
outputPath	- string
modeList	- string array

iface	- integer (any value for example 1)
FeatWght	- integer array
Wght_En	- integer (1- enable, 0-disable)
Std_En	- integer (1- enable, 0-disable)
distSel	- integer (2-Manhattan distance,1-Euclidian distance)
geoDist	- integer
geoDist_En	- bool (True - enable, False - disable)
mask	- string (Path to .TIF file)
mask_en	- integer (1- enable, 0-disable)