EO-Learn: A Basic Machine Learning Library for Land-Cover Classification

Chris Burgoyne

Introduction

Earth Observation (EO) is the gathering of data about the physical, chemical and biological systems of the earth. It can be performed by gathering data remotely or by using closer ground-based technologies. One common way that is used to gather EO data is by housing sensors on aircraft and satellite platforms, and recording data while moving aerially, or in low-earth orbit. It is a useful and cost-effective tool for gathering large amounts of data that cover vast swathes of the earths surface and atmosphere. It is also useful in reaching areas that would be difficult to record information in using land-based approaches (Lambin and Helmut, 2008).

While this data can take multiple formats, much of the data can be reduced to a numeric representation where information from various sensors is recorded to represent certain states. As such, this data lends itself well to mathematical and statistical analysis.

Land-cover classification is one of the tasks uses numeric analysis with remotely sensed data. It is an important task that has implications for many fields such as land management, urban planing, agriculture, and environmental protection (Tong et al. 2019). The aim of land-cover classification is to provide labels to describe the type of physical cover that land has (Helber et al. 2019). In the past, these labels have been applied by people examining aerial photos, marking them by hand, and using assessment techniques that involved visiting the areas being classified. However today, land-cover is changing more rapidly than ever before (Lambin and Helmut, 2008). As a result, up to date classifications of large land areas performed at regular intervals are necessary if this data are to remain applicable (Yifang et al. 2015). As such, automation of the classification process with remotely sensed data is a useful task.

Land-cover classification can be performed using a variety of statistical techniques that numerically automate the labeling of pixels in satellite imagery. These techniques are broadly classified as machine learning techniques for data analysis, where machine learning is considered a branch of artificial intelligence that includes systems which can learn from data, identify patterns, and make decisions with minimal human intervention (Zhang 2020).

In this paper, a basic machine-learning library is presented for the purpose of performing land-cover classification using satellite data. The library is written in the Python programming language and includes a variety of tools for working with and interpreting remotely sensed data that is in numeric format. Beyond an introduction of the library, this paper also contains an example in the application of the library tools with presentation of the data used, the methods employed, and the results from this investigatory process. The library is open source and can be found at github.com/koos-burgoyne/EOLearn.

The EO-Learn Library

The EO-Learn library is a basic Python library that can be used for applying machine learning techniques to the task of land-cover classification. The library contains four distinct parts:

- Data import and formatting
- Application of classification algorithms to numeric remote sensing data
- Evaluation of algorithmic performance
- Visualization of labeled data.

This library assumes that the data is within the desired geographic boundaries. It does not contain any functionality for performing geo-referencing or clipping data. Other Python libraries can be used for accomplishing this task, like the commonly used Geospatial Data Abstraction Library (GDAL).

Data Import and Formatting

The import and format module uses the Matplotlib Python library for importing image data. Each band from a particular image should be separate and stored in GeoTiff format. They should be labeled in sequential format according to which band they contain data for. The import_data function is then provided the directory where the images are stored and it imports these images into memory using a Matplotlib function. The images are assessed and returned for further use.

Other functions from the import and formatting module that are useful include functionality for:

- Retrieving features from the imported data
- Splitting the data into train and test datasets
- Resampling data:
  - Resampling randomly or resampling with a returned sample having balanced classes
  - Up-sampling where all classes are resampled to the size of the largest class
  - Down-sampling where all classes are resampled to the size of the smallest class
  - Combining classes where classes below the mean class size are combined to form larger unions of arbitrarily joined classes that were below the mean class size
- Removing instances of data that have a class value of 0 – this is useful because when an image extent includes areas that were not within a clipping mask, those pixels have a value of 0 and are thus not useful
- Provision of land-cover labels according the 2019 National Land-Cover Dataset that covers the land mass of the lower 48 states of the United States of America.

Classification

For classification functionality, the library has a module called ‘Algorithms’ that contains four machine learning techniques that can be applied for the purpose of classifying land-cover. The four algorithms implemented include:

- K Nearest Neighbours
- Naive Bayes
- Decision Tree
- Random Forest.

For an explanation of these algorithms see Ray, 2019. These algorithms are all used in a similar format: they are called using the ‘classify’ function, provided with training and testing data, and given a set of arguments (excepting the Naive Bayes classifier) that are hyperparameters for the respective model. While the K Nearest Neighbours algorithm makes use of the KD-Tree implementation from SciKit Learn[1] in order to save on computational expense, the others are implemented without using any external libraries. These algorithms can also be called using the evaluation module’s cross-fold validation function.

The reason that these algorithms were chosen is that the Random Forest and K-Nearest Neighbours remain commonly used algorithms for various land-cover classsification tasks (Thanh Noi and Kappas, 2018). In addition to the Random Forest, a single Decision Tree was thought to be a good benchmark for the Random Forest algorithm. Being faster to implement, one can get a quicker idea of how this type of classification might perform by using the Decision Tree before implementing the slower Random Forest. Lastly while Bayesian algorithms are less common, they do get used with a fair amount of regularity for land-cover classification (Pham, 2019). Moreover, for the purposes of this library a Naive-Bayes was an easier algorithm to implement that say a Neural Network. Settling on these four was decided as a fair benchmark for attempting to reproduce some existing and proven results.

<u>Evaluation</u>

The library also contains a module for evaluating classification work performed by the algorithms in the library. Functionality includes:

- Cross-fold evaluation which can be performed for either one of the in-library classifiers or for a SciKit Learn classifier[2]
- Splitting data into folds for cross-fold validation
- Returning the classes contained in a dataset
- Calculating a confusion matrix based on classification results and observed data labels
- Calculating error metrics from a provided confusion matrix
    - These include the True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR), False Negative Rate (FNR), Cohen’s kappa value, the Matthew’s Correlation Coefficient (MCC), and percentage accuracy.
- Printing the error metrics to comma separted value (.csv) format.

<u>Data Visualisation</u>

Lastly, the library provides functionality for visualising the data. This can include data with labels that are provided with the training and testing data, or data that has been labeled by machine learning classification techniques. There is also functionality for plotting counts and distributions of class values in labeled data. This functionality is provided by the Matplitlib Python library. Examples of the results from this module will be displayed throughout this paper.

1 http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KDTree.html
2 https://scikit-learn.org/stable/supervised_learning.html

Testing the Library

Data

The data used in this paper are provided by the European Space Agency Sentinel-2 Satellite. The Sentinels are a fleet of satellites designed specifically for providing data and imagery for the European Commision's Copernicus programme[3]. This programme aims to achieve global, continuous, autonomously gathered high quality remote sensing data of the earth. The Sentinel-2 satellites are two platforms in the same orbit 180° apart that carry high-resolution multispectral imagers with 13 spectral bands for earth surface monitoring. They have a swatch width of 290 kilometers and mainly provide information for agricultural and forestry applications.

The data was obtained from the United States Geological Service (USGS) Earth Explorer internet portal[4]. Two images were used in the evaluation of this library. The training and testing data was obtained from tile number T11PN while the validation data was obtained from tile number T11TQN. Both images have an acquisition date of July 22, 2021. They are both cloud free.

The labels for land cover classes come from the USGS 2019 National Land Cover Database (NLCD). The 2016 data was evaluated as having an overall accuracy of 86.4% (Wickham et al. 2021). The resolution is More information about this data can be found at the NLCD internet portal[5]. All data was put into the coordinate reference system WGS 84 / UTM zone 11N.

The Sentinel-2 images, along with the NLCD label data, were clipped using the GDAL Python library and two separate shapefiles, one for training/testind data and another for validation data. Detail on these can be found in the repository. At no point were these bands combined into a single raster. In concordance with the library requirements, they were stored as separate GeoTiff rasters. The resolution of all Sentinel-2 images was revised to 30m to match that of the NLCD images. The extent of these clipped images covers much of the Mission Valley, an area of land that is within the Flathead Reservation in the state of Montana, USA. As can be seen in Figure 1, the western half of the valley was used for training and testing data, while the eastern half along with the Mission Mountains was used for validation. There was slight overlap between the two extents.

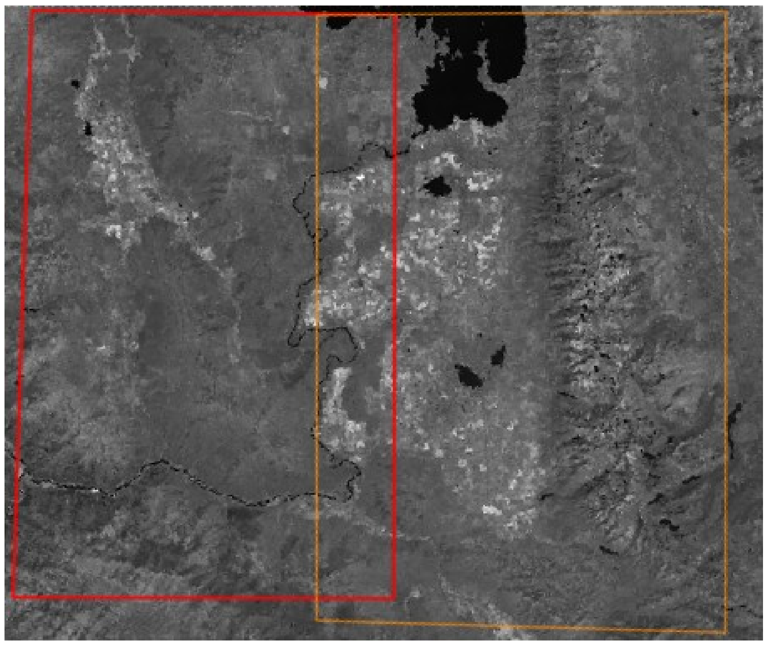

Figure 1: *A graphic representation of the Sentinel-2 imagery and the two areas used to clip the images, the red extent being for training and testing and the orange extent for validation of a developed model.*

3 https://www.esa.int/Applications/Observing_the_Earth/Copernicus/Sentinel-2/Introducing_Sentinel-2
4 https://earthexplorer.usgs.gov/
5 https://www.mrlc.gov/data/nlcd-2019-land-cover-conus

The Sentinel-2 data[6] included in this assessment include the bands numbered 2 (blue), 3 (green), 4 (red) and 8 (near-infrared). These bands were used because they are commonly used for vegetation assessment while restricting the testing to these bands also reduced the computational cost during testing, a repetitive process. As will be discussed later, this does place limitations on the results but the initial results were deemed good enough for the purposes of this investigation.

No atmospheric correction was performed during the pre-processing of the imagery. The purpose of pre-processing was merely to get the data into the correct format and extent for the purposes of the report.

Initial investigation of the data showed that the land-cover classes in these images were unbalanced. Figures 2 shows that the pixel counts for the various classes varied between a few hundred pixels and over a million. This extreme imbalance has the potential to skew the training of a model, thus resulting in poor classification results. As a result, various resampling techniques were experimented with.

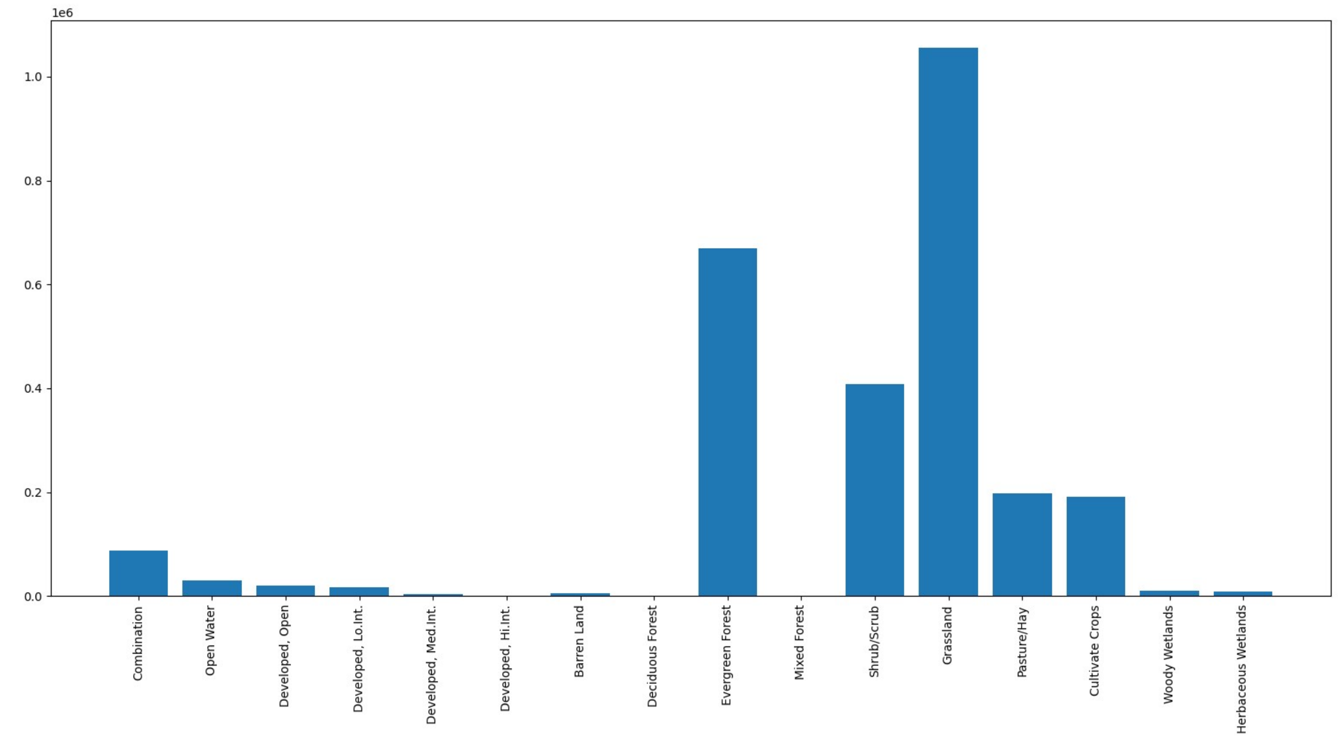

Figure 2: *The class counts for each of the classes contained in the training data. A similar plot of unbalanced classes was seen for the validation data.*

<u>*Methods*</u>

Following the acquisition and clipping of data, the library was then employed to classify the data and evaluate the results. To begin with, each image (all included bands of Sentinel-2 and the NLCD labeled data) was imported into memory as a two dimensional array, with each position *i*,*j* being a particular

6 https://www.usgs.gov/centers/eros/science/usgs-eros-archive-sentinel-2?qt-science_center_objects=0#qt-science_center_objects

pixel with row *i* and column *j* in the input image. These images were then converted to features that could be classified using a machine learning algorithm. The resulting format was another two dimensional array, this time each column being a flattened image so that the column contains every pixel from an input image. Each row represents a pixel with the various columns containing the information from each of the included Sentinel-2 bands. The last column was the land-cover labels of each pixel, recovered from the 2019 NLCD labeled data.

Following data import and formatting to features, the data can be sampled if desired, and then passed to a classification algorithm. In this case, a cross-fold validation technique (Refaeilzadeh et al. 2009) was implemented for classifier training with five equally sized folds. There were thus 5 iterations of training and testing where the *ith* fold would be reserved for testing and all other folds used for training. The classification results of these iterations were summed in a confusion matrix and the overall error metrics calculated for the entire process.

To test the different classification algorithms implemented, the training data was imported and formatted, resampled to 10% of the original sample to aid in fast testing, split into folds, and then classified sequentially by all four classifiers. These same data were subsequently passed to the equivalent SciKit learn classifiers, along with additions of the SciKit Learn implementations of the Support Vector classifier and a Multi-Layer Perceptron. The results were recorded for each classifier and the best performing classifier was chosen for subsequent steps. Classifier performance was evaluated using the error metrics described in the bullet points on page 4.

Following this, the training data was resampled using up-sampling and down-sampling techniques, along with finally combining classes where the class size was below the mean class size. The training data resulting from each of these sampling techniques was used to train the library implementation of the best performing classifier, with the best performing classifier then being used to classify a separate set of validation data that was described previously. The purpose of this was validation of the model produced from the training data, using separate data to determine how well the resulting model generalized to similar data. The results were recorded at each step, along with a visualisation of the classified validation data. The results of this process are reported below.

Results

After the three rounds of testing which compared a variety of SciKit-Learn classifiers to EO-Learn classifiers, the error metrics from Table 1 were recorded. The general trend in error metrics can be seen in Figure 3, which shows the percentage accuracy of the various classifiers used in this study. The graphs of the other metrics, which follow a similar distribution of values as seen in Figure 3, can be found in Appendix A. The three rounds of testing involved performing classification using the various classifiers listed in Table 1 on firstly the unbalanced training data, then resampling that involved balancing the class counts, and finally a resampling that involved combining classes where the class size was below the mean of class sizes.

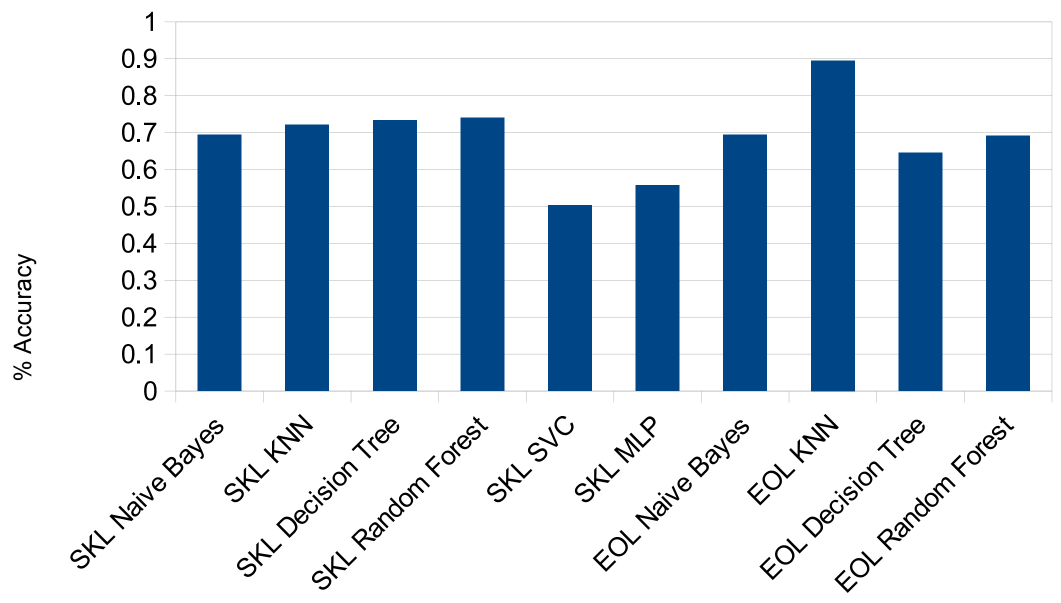

Figure 3: A plot of the percentage accuracy across all tested classifiers.

When performing the testing on training data that was unbalanced across the classes, the percentage accuracy was relatively high. The range of values was from 35% for the SciKit-Learn Multi-Layer Perceptron to 85% for the EO-Learn KNN classifier. When testing on the training data that was balanced across the classes (for the various balancing techniques mentioned above), the percentage accuracy was worse than the accuracy on the unbalanced dataset. This was true for all the classifiers. When comparing the error metrics across testing with unbalanced and balanced datasets, the unbalanced dataset produced worse TPR and FPR values. This makes it apparent that while the accuracy was better for the unbalanced testing data, the reason was that it was classifying most of the data as the largest classes. As a result, the classification will not be particularly valuable.

In the above two test iterations, the distribution of the metrics closely followed the distribution seen in Table 1. As such, there was no reason to repeat the table across the various tests (as well as for the sake of space) but just to illustrate, when working with the balanced training data the accuracy rate for the EO-Learn KNN was 55.8% while for the SciKit-Learn KNN it was 47.5%.

When performing this same testing on training data that was created from combining classes with a size below the mean class size, the error metrics increased across the board and can be seen in Table 1.

From the error metrics observed in testing, the EO-Learn KNN classifier was deemed the best classifier for this dataset. The justification for this final choice was that while the runtime was not the best, it consistently performed the best in all of the error metrics: The Kappa score, percent accuracy, and mean TPR were highest for the EO-Learn KNN, while it also had the lowest mean FPR. The distribution of these results was mirrored across a variety of sizes of resampled datasets, from 1% to 50% of the size of the full testing dataset. As a result, the EO-Learn KNN classifier was chosen to perform the validation classification.

Table 1: *The results of testing using the class-combination training data, a test which involved a comparison of SciKit-Learn classifiers to EO-Learn classifiers.*

Classifier	Cohen's Kappa	MCC	% Accuracy	Mean True Positive Rate	Mean False Positive Rate	Runtime (seconds)
SKL Naive Bayes	0.554	0.566	69.4	0.643	0.12	0.064
SKL KNN	0.603	0.607	72.1	0.673	0.1125	2.229
SKL Decision Tree	0.617	0.625	73.3	0.69	0.105	0.551
SKL Random Forest	0.626	0.635	74.0	0.703	0.1025	4.622
SKL SVC	0.383	0.467	50.3	0.398	0.0675	41.198
SKL MLP	0.414	0.526	55.8	0.304	0.0454	5.176
EOL Naive Bayes	0.554	0.566	69.4	0.643	0.12	16.134
EOL KNN	0.852	0.857	89.5	0.873	0.038	6.557
EOL Decision Tree	0.505	0.616	64.5	0.582	0.073	67.551
EOL Random Forest	0.559	0.665	69.1	0.625	0.06	614.194

Following the initial rounds of testing, the validation data was then used for classification using the EO-Learn KNN classifier. Again, the classes smaller than the mean class size were combined as this provided the best results in initial testing. In Figure 3, the NLCD labeled data (seen on the left) was also passed through the same filter to combine classes smaller than the mean class size. This was then used for calculating the error metrics of the validation classification. The validation data was classified and the results can be seen in Table 2 and Figure 3.

As seen in Table 2, the EO-Learn KNN classifier was applied to the validation dataset across four values of K. What we see is that the error metrics gradually worsen with an increasing value of K. The classification produces a fairly similar result to the observed labels, which is fairly impressive given the model was trained on a separate image. Some obvious differences were apparent but a fairly good representation of the original labeling being achieved by the EO-Learn KNN classifier.

Table 2: *The error metrics across 4 values of K for the EO-Learn KNN performed on validation data.*

K	% Accuracy	Kappa	MCC	Mean TPR	Mean FPR
3	62.8	0.504	0.522	0.540	0.110
7	60	0.467	0.482	0.507	0.123
20	60	0.467	0.482	0.507	0.123
30	55	0.397	0.409	0.445	0.150

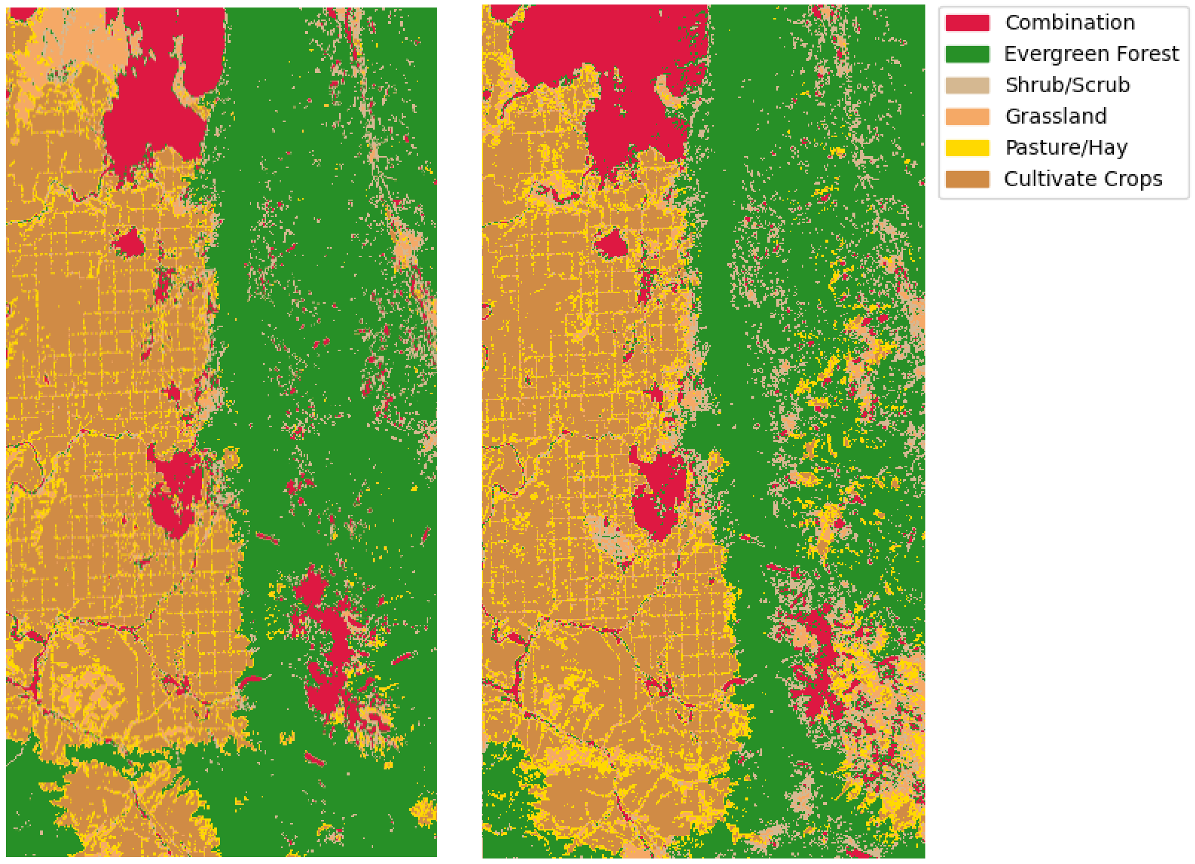

Figure 4: *A side by side comparison of the NLCD labeled data (on the left) and the EO-Learn KNN classified data (on the right). The accuracy of the EO-Learn classification on a class-combination resample of the validation dataset was 62.8% with K=3.*

<u>Discussion</u>

The worsening error metrics for the increasing value of K seen in Table 2 indicates that the data is closely clustered in the numeric values that appear across classes. This is illustrated in Appendix A Figures 9 and 10. These figures show the distribution of earth surface reflectance values for each of the bands (Figure 9) and each of the classes in Band 1 as an example (Figure 10). What these figures do is show that the data in all of the classes closely resemble one another. This was similar across the bands. Naturally we would assume that the more closely the data is clustered, the harder it will be to differentiate classes. This would indicate one reason why the performance was not particularly good when classifying on data from a separate image.

Considering this information about the close clustering of values across the classes, it seems amazing that the KNN classifier can even classify the data with the accuracy that it does. However that is what makes autonomous classification through machine learning so powerful. Where patterns are not at first apparent to humans, the powerful statistical techniques available to us (that can be computerised and evaluated in seconds) can allow use to classify the numeric data measured from hundreds of kilometers above the earth’s surface with relatively good accuracy.

What we also see in Appendix A Figure 9 is that the data across the bands is fairly closely clustered. That is perhaps one major weakness of this testing. While the four selected bands were included to improve the runtime, the small number of bands likely had an impact on the error metrics. There are six other bands included in the Sentinel-2 images that could be included when training, testing, and validating models[6]. This would likely increase the distribution of feature values across classes, improving the classifiers ability to classify each pixel with the correct class. Future testing might involve such an approach.

Another reason for the less than perfect performance could be that the NLCD data used for labeling (and thus calcualting error metrics) does not have perfect accuracy. As a result, the model will be confused during training by pixels that have been classified incorrectly. This would worsen the performance of the resulting classifier.

Moreover the NLCD created using data from the Landsat satellite sensors while this study was performed using data from the ESA Sentinel-2 system. If the data source was matched with the data source that produced the labeling, there might be a better correlation in the labels and the classification result.

Conclusion

This paper presented a small machine learning library for the purpose of classifying land-cover from satellite images. The overall result was promising, with a similar classification occurring when classifying on the same image that a model was trained on. Although the result obtained from classifying a different image to the training image was less accurate, it was percievably similar to the original classification. The objective of producing a classified land-cover image that was similar to the NLCD image was successfully met, but the accuracy could be improved by ensuring certain conditions are met with regard to the data.

As to why the EO-Learn classifier performed the best remains a mystery. Overall, it was a surprise that the KNN classifiers were the best performers, however it might be the lack of numeric complexity that has caused the other classifiers to have a poorer performance. It seems likely that the performance as reported by the error metrics will change when classifying with data that has more features (inclusion of more of the available bands from the image). Further testing incoporating the above-mentioned changes could reveal different results.

References:

Helber, Patrick, Benjamin Bischke, Andreas Dengel, and Damian Borth. "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12, no. 7 (2019): 2217-2226.

Lambin, Eric F., and Helmut J. Geist, eds. *Land-use and land-cover change: local processes and global impacts*. Springer Science & Business Media, 2008.

Pham, Binh T., Indra Prakash, Khabat Khosravi, Kamran Chapi, Phan T. Trinh, Trinh Q. Ngo, Seyed V. Hosseini, and Dieu T. Bui. "A comparison of Support Vector Machines and Bayesian algorithms for landslide susceptibility modelling." *Geocarto International* 34, no. 13 (2019): 1385-1407.

Ray, Susmita. "A quick review of machine learning algorithms." In *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)*, pp. 35-39. IEEE, 2019.

Refaeilzadeh, Payam, Lei Tang, and Huan Liu. "Cross-validation." *Encyclopedia of database systems* 5 (2009): 532-538.

Thanh Noi, Phan, and Martin Kappas. "Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using Sentinel-2 imagery." *Sensors* 18, no. 1 (2018): 18.

Tong, Xin-Yi, Gui-Song Xia, Qikai Lu, Huanfeng Shen, Shengyang Li, Shucheng You, and Liangpei Zhang. "Land-cover classification with high-resolution remote sensing images using transferable deep models." *Remote Sensing of Environment* 237 (2020): 111322.

Wickham, James, Stephen V. Stehman, Daniel G. Sorenson, Leila Gass, and Jon A. Dewitz. "Thematic accuracy assessment of the NLCD 2016 land cover for the conterminous United States." *Remote Sensing of Environment* 257 (2021): 112357.

Yifang, Ban, Peng Gong, and Chandra Gini. "Global land cover mapping using Earth observation satellite data: Recent progresses and challenges." *ISPRS journal of photogrammetry and remote sensing (Print)* 103, no. 1 (2015): 1-6.

Zhang, Xian-Da. "Machine learning." In *A Matrix Algebra Approach to Artificial Intelligence*, pp. 223-440. Springer, Singapore, 2020.

Appendix A

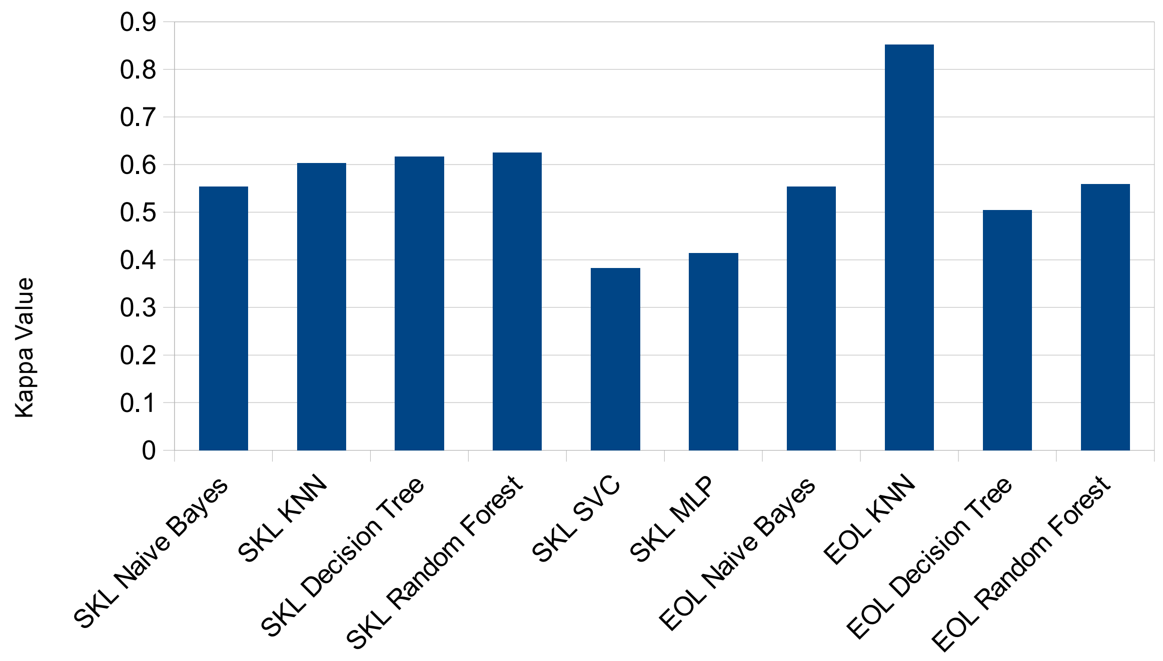

Figure 5: A plot of the value for Cohen's Kappa across all tested classifiers.

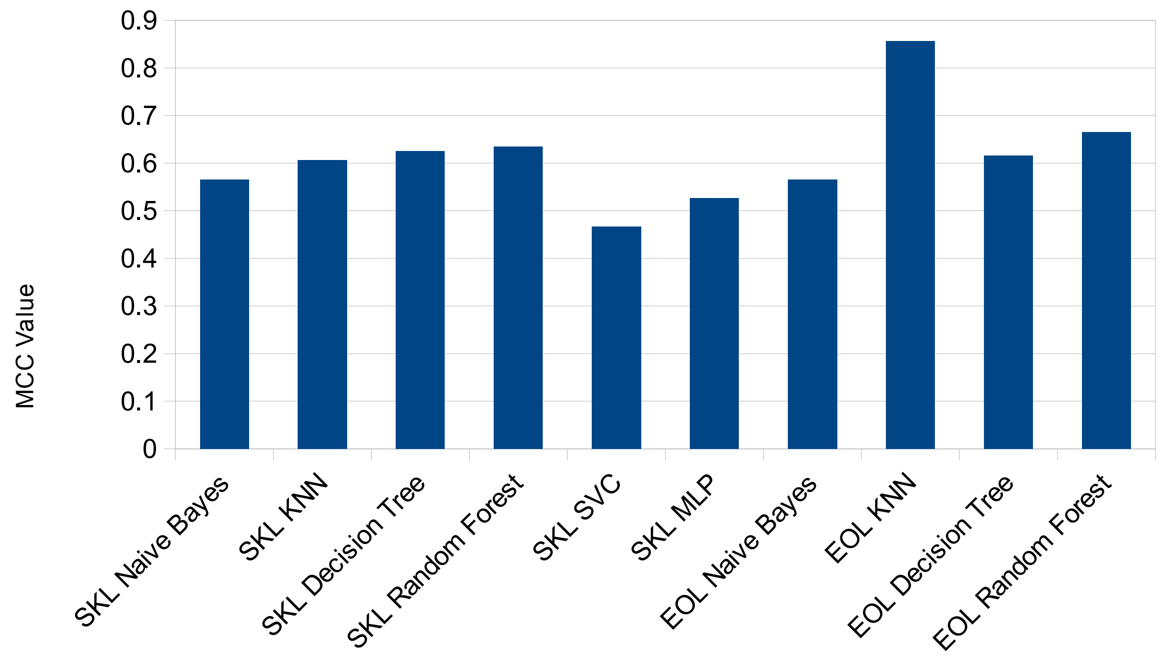

Figure 6: A plot of the Matthew's Correlation Coefficient across all tested classifiers.

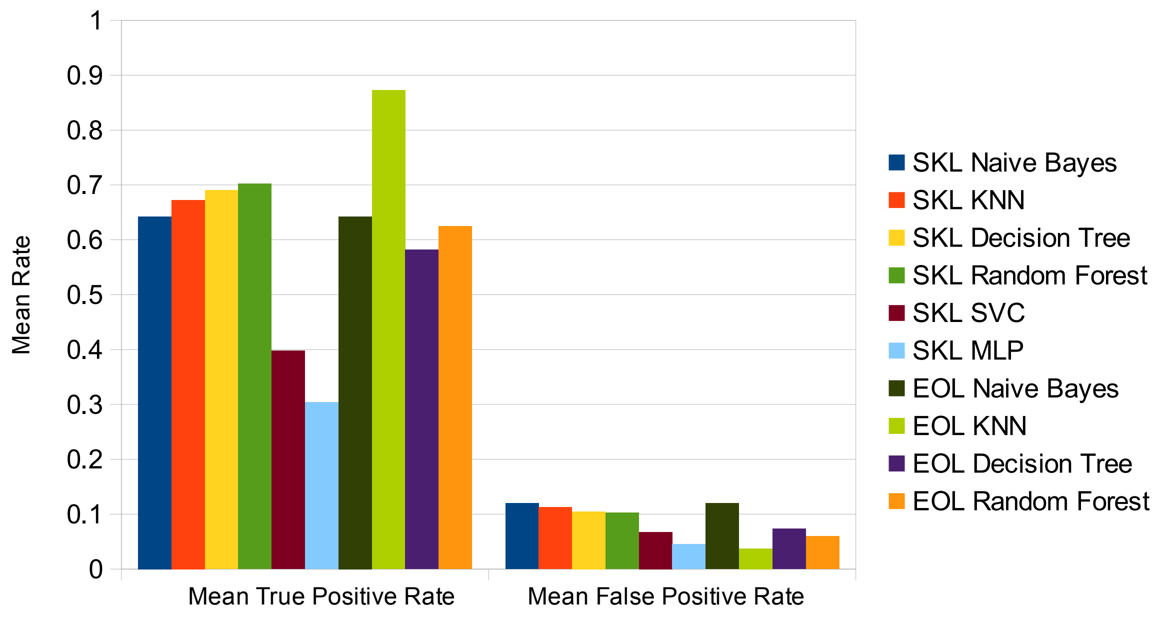

Figure 7: A plot of the mean TPR and mean FPR across all tested classifiers.

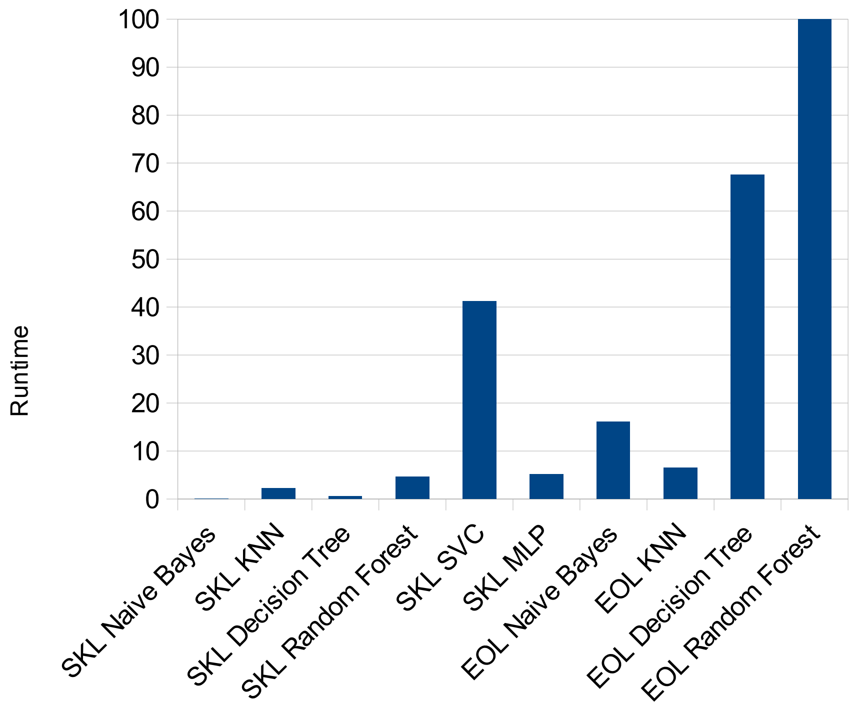

Figure 8: A plot of the average runtimes (16 replicates) for all tested classifiers.

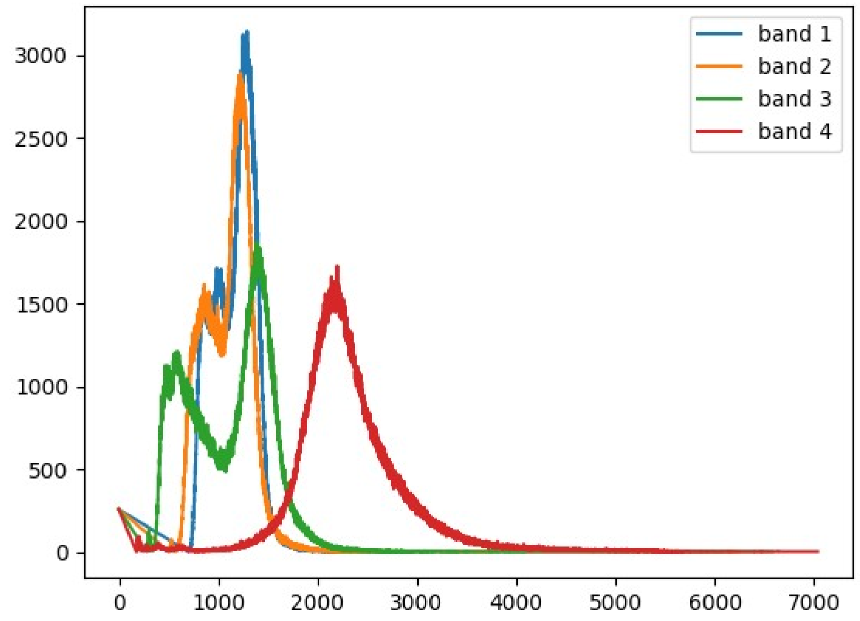

Figure 9: A plot of the frequency of numeric values appearing in the 4 Sentinel-2 bands used, as counted in a sub-sample of the training dataset. Notice the close distribution of values, as well as the similar frequencies.

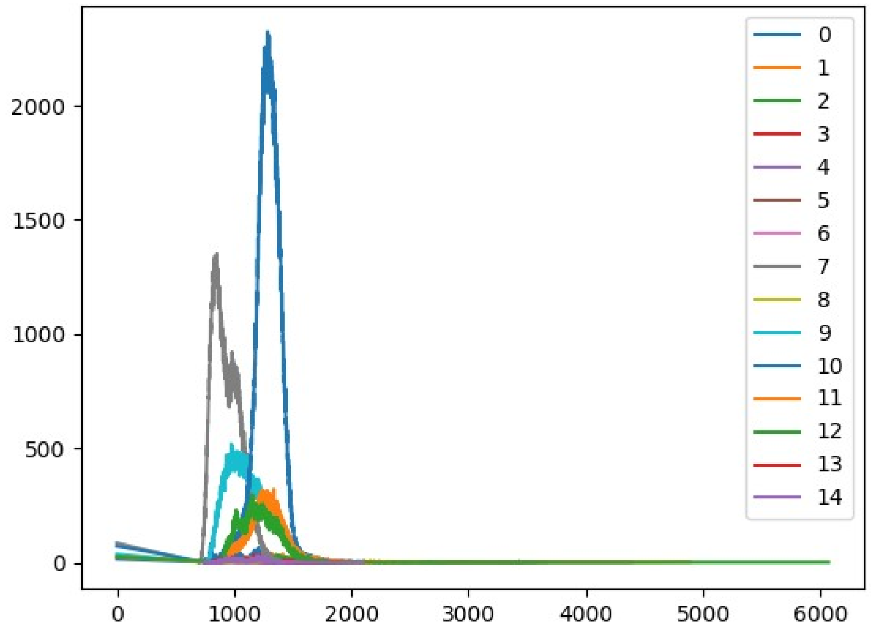

Figure 10: A plot of the frequency of the Band 1 numeric values across all 15 classes represented in a sub-sample of the training dataset. Notice the close distribution of values, as well as the similar frequencies.