****

ANKARA YILDIRIM BEYAZIT UNIVERSITY

Department of Computer Engineering

**Project Fall**

Literature Research

**Generating Land Cover Maps for Gibraltar**

**using Limited Training Samples**

Student name: Orçun *TURKOKULOGLU - 18050161002*

Course: *Machine Learning (CENG 463)*

Instructor: *Dr. Mustafa TEKE, Computer Engineering*

Due date: *February 25th, 2023*

## **Abstract**

In this thesis, a comprehensive literature review was conducted on the topics of machine learning and artificial neural networks. The specific focus of the research is the generation of land cover maps for unseen data utilizing a limited number of training samples, with the study area being Gibraltar due to its diverse array of land cover types.

[Land Use Land Cover Mapping Using Advanced Machine Learning Classifiers](https://sciendo.com/pdf/10.2478/eko-2021-0031)

**Introduction**

In this report, I aim to generate land cover maps for an unseen area using a limited number of training samples. The study area chosen for this research is Gibraltar, as it has a wide variety of land cover types. I will be using Sentinel-2 data provided by ESA and implementing machine learning algorithms, such as Random Forest, KNN, and K-Means, to classify the land cover in the study area. The main focus of this research is to use limited training samples and classify unseen data to understand the impact of climate change on land cover. The ultimate goal is to contribute to the preservation of land cover and tackle the problem of climate change.

## **Table Of Contents**

[**Abbreviations**](#_z271yr5m9en5) **3**

**Land Use Land Cover Mapping Using Advanced Machine Learning Classifiers 4**

[**People**](#_3znysh7) **4**

[**Requirements**](#_2et92p0) **4**

[**Research Problem**](#_auqf83e4s23y) **5**

[**Novelty/Contribution**](https://docs.google.com/document/d/1u1vaBzZB2vlgnXrrR238-__n9s5Rw-z-2g7RrRUD7Gk/edit#heading=h.r3usxwfomhd2) **6**

[**Impact**](https://docs.google.com/document/d/1u1vaBzZB2vlgnXrrR238-__n9s5Rw-z-2g7RrRUD7Gk/edit#heading=h.wi4faq1bfg0) **6**

[**Reference**](#_5f7iuvqf909m) **9**

[**Figures**](#_pmxybvezu0z0) **10**

[**Tables**](#_2sclhjefdtkf) **10**

| **Abbreviations** | |
| --- | --- |
| **LULC** | Land Use Land Cover |
| **SVM** | Support Vector Machine |
| **RV** | Random Forest |
| **MLP** | Multi-layer Perceptron |
| **GA** | Genetic Algorithm |
| **OA** | Overall Accuracy |
| **Sentinel-8** | A Satellite Data |
| **Sentinel-2** | A Satellite Data |
| **MATLAB** | Matrix Laboratory |
| **K-NN** | K - Neighbours |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# **Land Use Land Cover Mapping Using Advanced Machine Learning Classifiers**

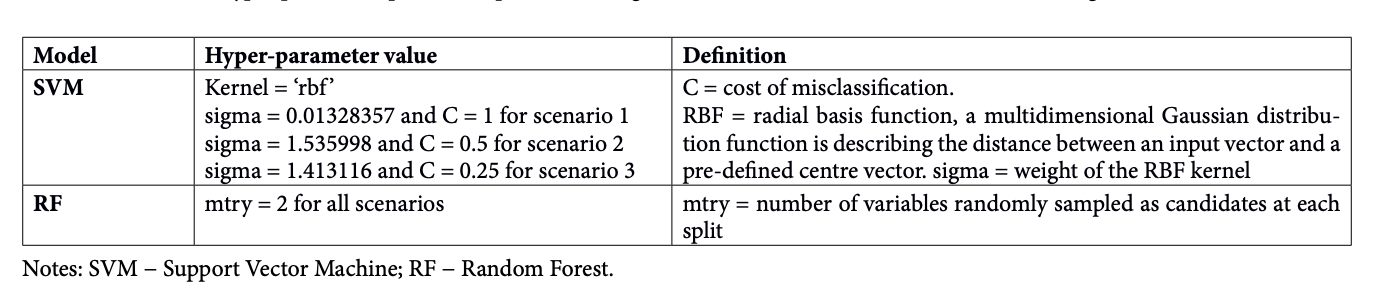
## **People**

ALI JAMALI

* Dr. Ali JAMALI is an Instructor Member at the engineering faculty of Civil Engineering. He received his BSc degree from Universiti Teknologi Malaysia in 2012 and completed it in 2017. His research areas include Remote Sensing, Photogrammetry, Cadastre, and Property. He has a total of 49 publications in these areas. Dr. JAMALI has a wealth of knowledge and experience in these fields and is a valuable asset to the engineering faculty.

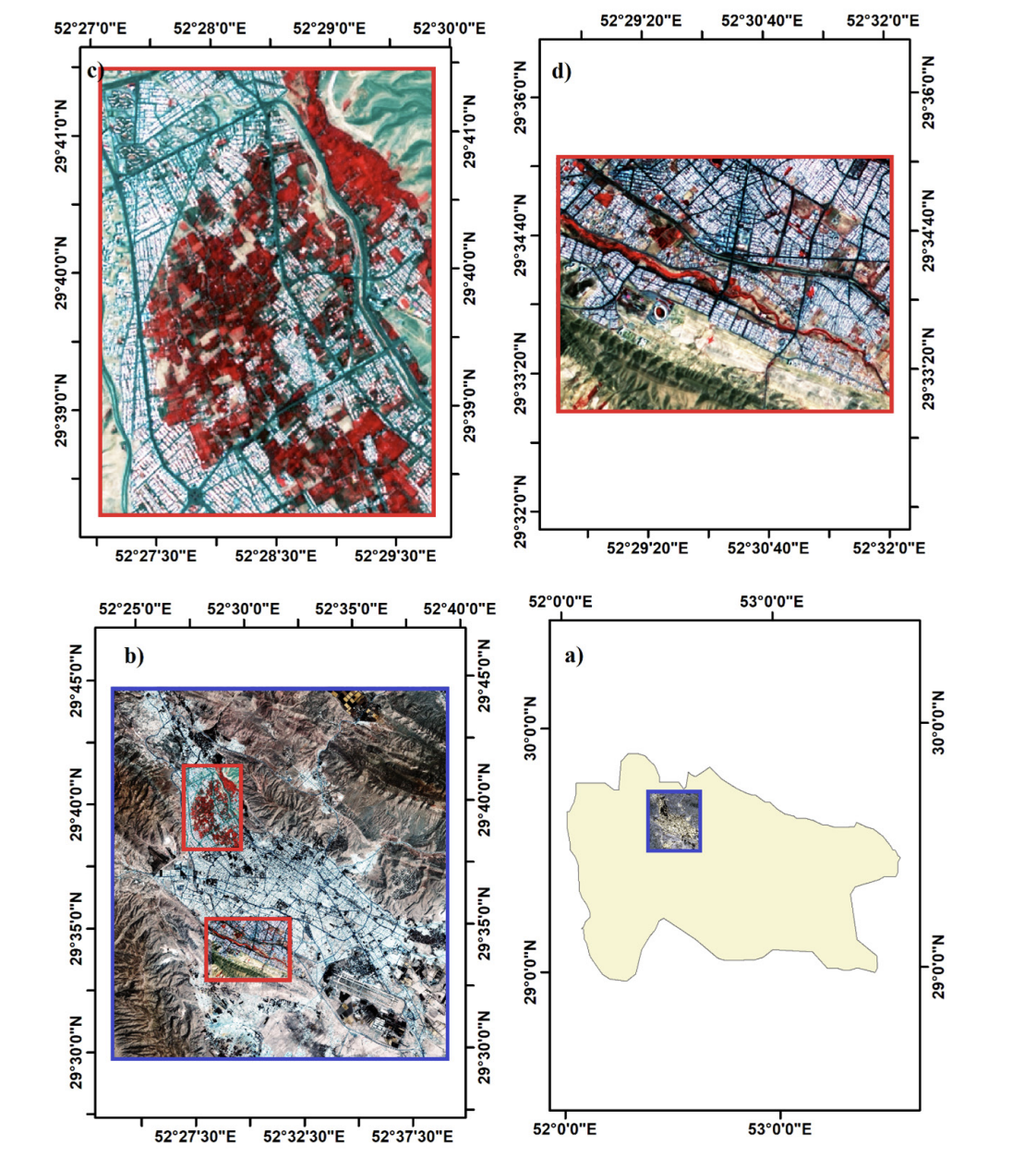
## **Requirements**

In the last decade, remote sensing and data science researchers have used advanced machine learning algorithms for remote sensing image classification (Jamali, 2019; Mahdianpari et al., 2017, 2019; Rodriguez-Galiano et al., 2012; Rogan et al., 2008; Shao, Lunetta, 2012; Yeom et al., 2013). Free access satellite data such as Landsat-8 and Sentinel-2 has raised the use of image classification algorithms towards remote sensing field (Belward, Skøien, 2015; Harris, Baumann, 2015). On the other hand, the computing power of personal computers has been increasing, while its cost is decreasing at a rapid rate (Waldrop, 2016).



**Table 1. 1 SVM; RF**

In the Land Use Land Cover (LULC) mapping as a sub-field of image classification, the use of advanced machine learning algorithms has gained rapid interest (Jamali, 2020a,b,c; Jamali et al., 2021a,b) information on the Land Use Land Cover (LULC. For the physical and human environment, precise and up-to-data LULC dada is a need (Jamali, 2019), where it can be used in several fields, including health, ecology (Bourgeois, Sahraoui, 2020; Kenderessy et al., 2020; Skalský et al., 2018). Random Forest (RF) algorithm (Breiman, 2001) is considered as one of the most popular tree-based machine learning algorithms for image classification due to its simplicity and the fact that it can be used for both classification and regression problems with both continuous and categorical data (Woznicki et al., 2019). RF algorithms have been used in diverse fields such as land use modelling (Araki et al., 2018), forest cover mapping (Betts et al., 2017), land cover mapping (Nitze et al., 2017) and object-oriented mapping (Kavzoglu, 2017). On the other hand, Support Vector Machine (SVM), due to its capability to generalise complex objects, has outperformed other machine learning classifiers in various researches (Mountrakis et al., 2011; Shao, Lunetta, 2012). For example, Goodin et al. (2015) used six land use classes to classify Landsat-8 satellite images with the use of the SVM classifier, reaching a high overall accuracy (OA) of 88%. Besides, Mansaray et al. (2020) used SVM and RF classifiers to map paddy rice in China in 2015 and 2016. In their study, SVM and RF obtained high accuracies of 90.8 and 89.2%, respectively, for 2015 from Landsat-8 and Sentinel-1A images. For 2016, RF and SVM classifiers had high accuracies of 95.2 and 93.4%, respectively, from Landsat-8, Sentinel-1A and Sentinel-2A. Additionally, from the late 1980s, neural networks have been intensively used for image classification in remote sensing, where they are utilised in open-source and commercial remote sensing software (Mas, Flores, 2008).



**Fig. 1. The study area. a) Fars province; b) Shiraz city; c) region a in false color; d) region b in false color.**

Recently, deep learning classifiers are employed in various fields including land cover mapping (Li et al., 2016), crop mapping (Kussul et al., 2017), plant disease mapping (Mohanty et al., 2016), and oil palm tree mapping (Li et al., 2017). In this research, besides, two well-known machine learning algorithms, including SVM and RF classifiers, two adVol. 40, No. 3, p. 286–300, 2021 doi:10.2478/Eko-2021-0031 287 advanced classifiers, namely, the derivative-free function of FminSearch Multi-Layer Perceptron (FSMLP) and Genetic Algorithm Multi-Layer Perceptron (GAMLP), are developed in the MATrix LABoratory (MATLAB) programming language. In addition, results are compared in terms of OA and kappa index, and they are interpreted visually for various regions of the study area.

## **Research Problem**

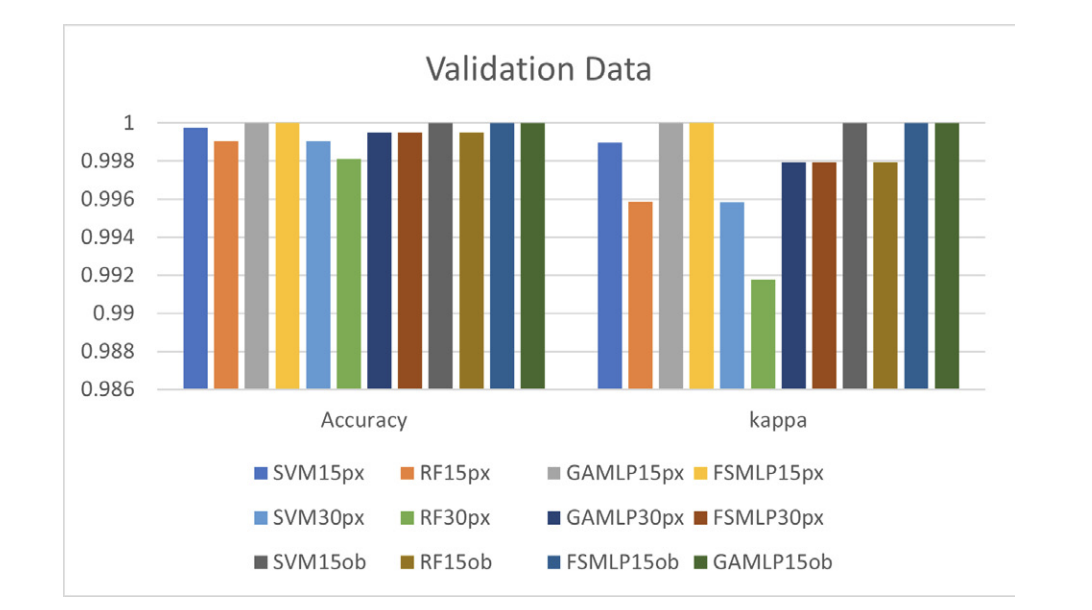
This study investigates the use of machine learning and deep learning algorithms for land cover mapping. Two popular classifiers, Support Vector Machine (SVM) and Random Forest (RF), were used, as well as two advanced algorithms, GAMLP and FSMLP, which are based on Multi-layer Perceptron (MLP) and optimized using Genetic Algorithm (GA) and a derivative-free function respectively. Three different scenarios using Landsat-8 imagery with different spatial resolutions were analyzed to evaluate the effect of data pre-processing on the final predicted land cover map. The results show that the developed MLP-based algorithms have high accuracy with over 98% correct classification. However, when compared visually and statistically, the GAMLP and FSMLP had the best results for pre-processed imagery with a 15m resolution but performed worse for unprocessed imagery compared to SVM and RF.

## **Novelty/Contribution**

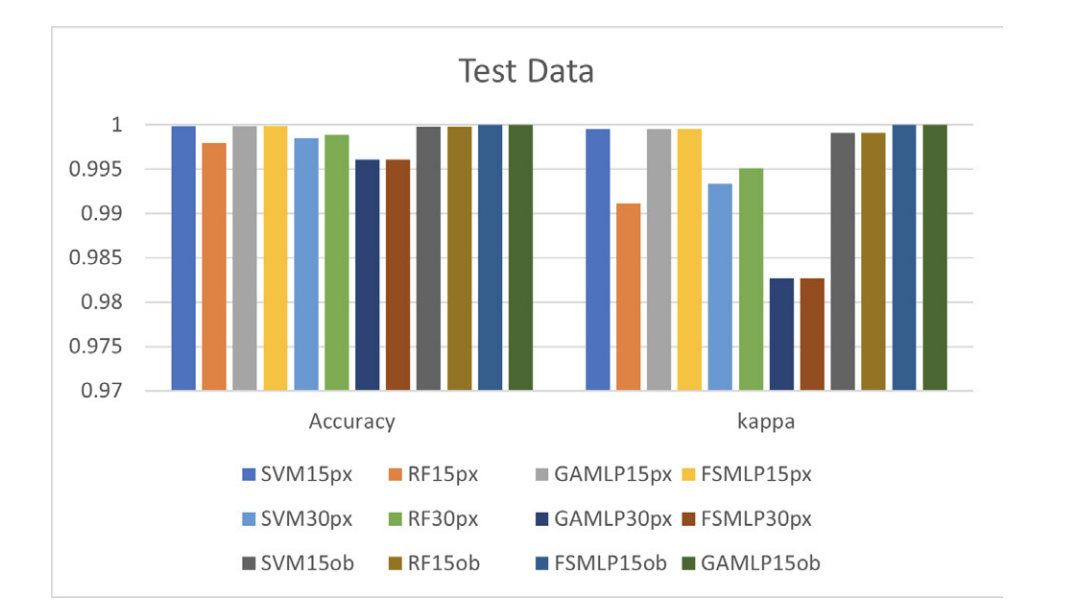
This study presents a novel approach to land use land cover (LULC) mapping using advanced machine learning algorithms, including Support Vector Machine (SVM) and Random Forest (RF), as well as two newly developed deep learning algorithms, the Genetic Algorithm Multi-Layer Perceptron (GAMLP) and FminSearch Multi-Layer Perceptron (FSMLP). The study also investigates the effect of data pre-processing on the final predicted LULC map using three different scenarios with Landsat-8 imagery. The results of this study show that the developed MLP-based algorithms have high accuracy with more than 98% correct classification, and offer a valuable contribution to the field of LULC mapping.

## **Impact**

The impact assessment of this research is based on the utilization of machine learning algorithms for land use and land cover (LULC) mapping in the Shiraz city, Iran. The study aims to investigate the effectiveness of different image pre-processing techniques and machine learning algorithms for LULC mapping. The research utilizes multi-spectral Landsat-8 image and applies three different scenarios for image pre-processing: atmospherically corrected and pan-sharpened to a spatial resolution of 15 m, without any correction at a spatial resolution of 30 m, and atmospherically corrected and pan-sharpened to a spatial resolution of 15 m with segmentation.

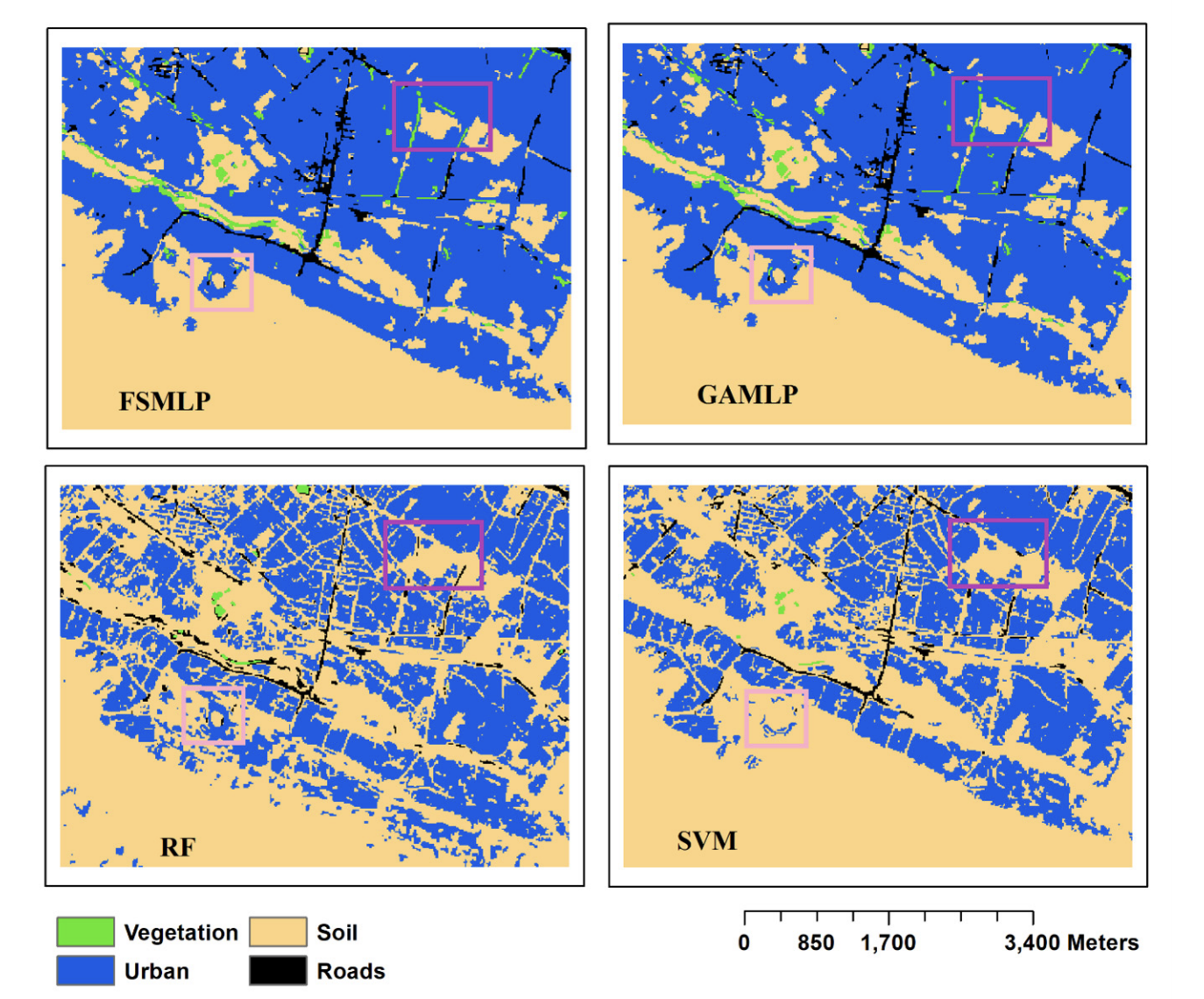


**Fig. 2. Results of machine learning algorithms for the validation dataset. 15px are images in scenario 1, 15ob are images in scenario 3 and 30px are images in scenario 2.**



**Fig. 3. Results for test dataset: 15px (scenario 1), 15ob (scenario 3), and 30px (scenario 2) images.**

The research applies two machine learning algorithms, Support Vector Machine (SVM) and Random Forest (RF), for LULC mapping and optimizes their performance through hyper-parameter tuning. Additionally, two optimized machine learning algorithms, Genetic Algorithm Multi-layer Perceptron (GAMLP) and fminsearch Multi-layer Perceptron (FSMLP) are developed based on Multi-layer Perceptron (MLP) function. The performance of the developed algorithms is evaluated based on the overall accuracy (OA) and the kappa coefficient (Kappa).



**Fig. 4. Final maps of the study area in region b for scenario 2 with a 30 m resolution using machine learning classifiers.**

The results show that the GAMLP algorithm performs better than the other algorithms for LULC mapping in Shiraz city. The study also demonstrates that the atmospherically corrected and pan-sharpened image with segmentation is the best suited data for LULC mapping in the study area. The research provides valuable insights for LULC mapping in similar urban areas and helps with the preservation of the garden regions in the study area by providing a machine-learning algorithm for monitoring the city. The research also highlights the importance of image pre-processing and the selection of appropriate machine learning algorithms for LULC mapping.

|  | K-NN | Bayes | RF | K-Means | Boosted Gradient | SVM  Classifier |
| --- | --- | --- | --- | --- | --- | --- |
| Cross Validation | 86.60% | 83.98% | 88.82% | -4907636244938.29% | 89.01% |  |
| Accuracy | 99.87% | 83.95% | 99.97% | 0.14% | 89.77% | 88.75% |

**Table 1.2 Cross validation accuracy and accuracy of methods**

**K-Neighbors Classifier FOR TRAIN.CSV**

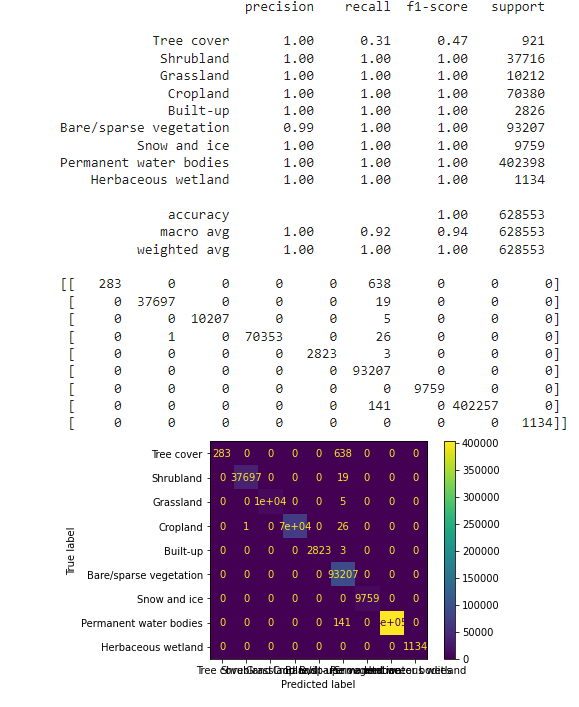
The k-nearest neighbors (KNN) algorithm is a supervised machine learning algorithm that can be used for classification and regression tasks. The idea behind the KNN algorithm is to find the k-number of closest training examples in the feature space for a new observation and predict the output of the new observation based on the majority class or average of the k-nearest neighbors.

In the classification task, the KNN algorithm assigns the new observation to the class that is most common among its k-nearest neighbors. In the regression task, the KNN algorithm predicts the output of the new observation as the average of the outputs of its k-nearest neighbors.

The KNN algorithm is simple to implement and easy to understand, but it can be computationally expensive as the number of observations increases. It also requires a large amount of memory to store the training data and it can be sensitive to irrelevant or noisy features.

Then, I will use the k-neighbors algorithm to classify new data points based on their proximity to labeled training data. The algorithm works by first selecting a number of nearest neighbors (k) to the new data point, and then analyzing the majority class among those nearest neighbors to classify the new data point. This algorithm is simple and easy to implement, but can be sensitive to the choice of k and the presence of outliers in the training data.

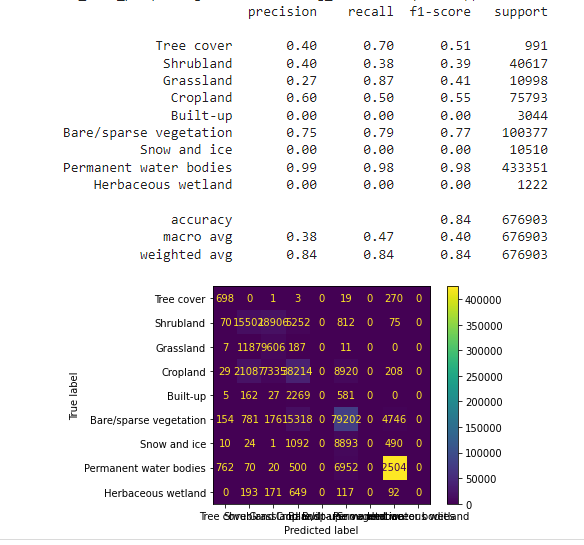
And I set the n value to 1, but the algorithm will only take into account the closest point to the new data point when making a prediction. This can lead to overfitting and a lack of generalization in the model. It's important to tune the n\_neighbors value and find the optimal number of neighbors to consider in order to achieve the best performance. So I needed to find the optimal number of neighbors and One way to find the optimal number of neighbors is to use cross-validation. I can use cross-validation to find the optimal number of neighbors for my K-Neighbors Classifier algorithm. I can start by trying different values for the number of neighbors, such as 1, 3, 5, 7, and so on. Then, I can use a technique such as k-fold cross-validation to evaluate the performance of my model with each value of the number of neighbors. The value of k that results in the highest accuracy on my validation set is likely to be the optimal number of neighbors for my model. Additionally, I can use techniques such as grid search or randomized search to systematically try different combinations of hyperparameters, including the number of neighbors, to find the optimal set of parameters for my model.



**BAYES METHOD FOR TRAIN.CSV**

In Bayesian classification, I start by assuming a prior probability distribution for the class labels, and then update this distribution based on the data that I observe. The basic idea is that I use Bayes' theorem to compute the posterior probability of the class label given the observed data, and then make a prediction based on the highest posterior probability.

* I define a prior probability distribution for the class labels. This can be based on prior knowledge or estimated from the training data.
* I estimate the likelihood function for each class, which represents the probability of observing the data given the class label. This can be done using techniques such as maximum likelihood estimation.I make a prediction based on the class label with the highest posterior probability.
* I can use the cross-validation method to find the optimal number of neighbors and re-train the model with the optimal number of neighbors.
* I can use the confusion matrix to evaluate the performance of the model.



**RANDOM FOREST CLASSIFIER FOR TRAIN.CSV**

Random forest is an ensemble method that utilizes multiple decision trees to make predictions. The idea behind this method is to combine the predictions of multiple decision trees to improve the overall accuracy of the model. This is achieved by creating a large number of decision trees, each one trained on a different subset of the data and with different subsets of the features. The final predictions are made by averaging the predictions of all the decision trees.

1. Method, select a random subset of the training data and create a decision tree model using this subset.
2. Repeat step 1 for 100 times to generate multiple decision trees.
3. For each decision tree, record the accuracy of the model on the validation data.
4. For new data, use the decision trees to make predictions and average the predictions to get the final prediction for the random forest model.
5. Select the random forest model with the highest accuracy on the validation data as the final model.
6. I use the final model to make predictions on new unseen data.

The advantages of random forest are that it is easy to use, it can handle large datasets, it can handle categorical and numerical variables, and it can handle missing data. The disadvantage is that it may be computationally expensive to train and test many decision trees.

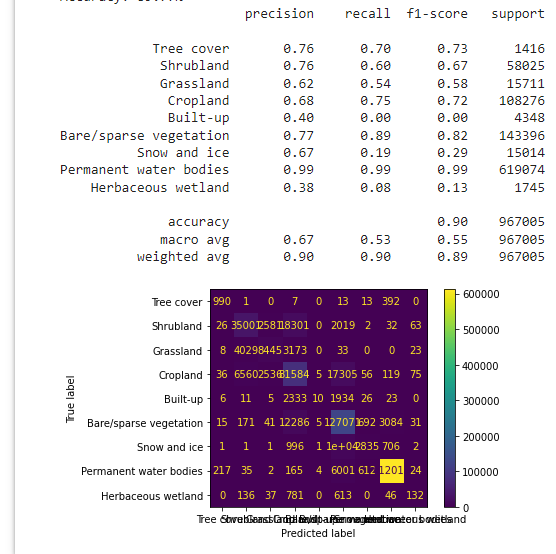


**Boosted Gradient For Train.Csv**

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.

Usage

Gradient boosting can be used in the field of learning to rank. The commercial web search engines Yahoo[16] and Yandex[17] use variants of gradient boosting in their machine-learned ranking engines. Gradient boosting is also utilized in High Energy Physics in data analysis. At the Large Hadron Collider (LHC), variants of gradient boosting Deep Neural Networks (DNN) were successful in reproducing the results of non-machine learning methods of analysis on datasets used to discover the Higgs boson.[18] Gradient boosting decision tree was also applied in earth and geological studies – for example quality evaluation of sandstone reservoir.[

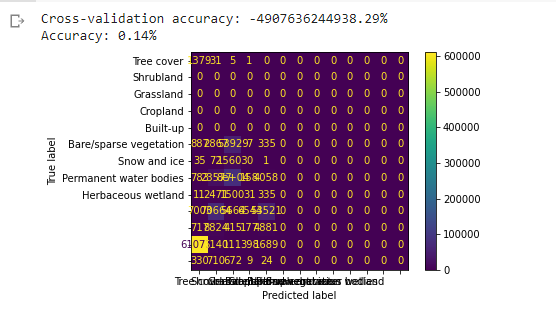


**K-Means For Train.CSV**

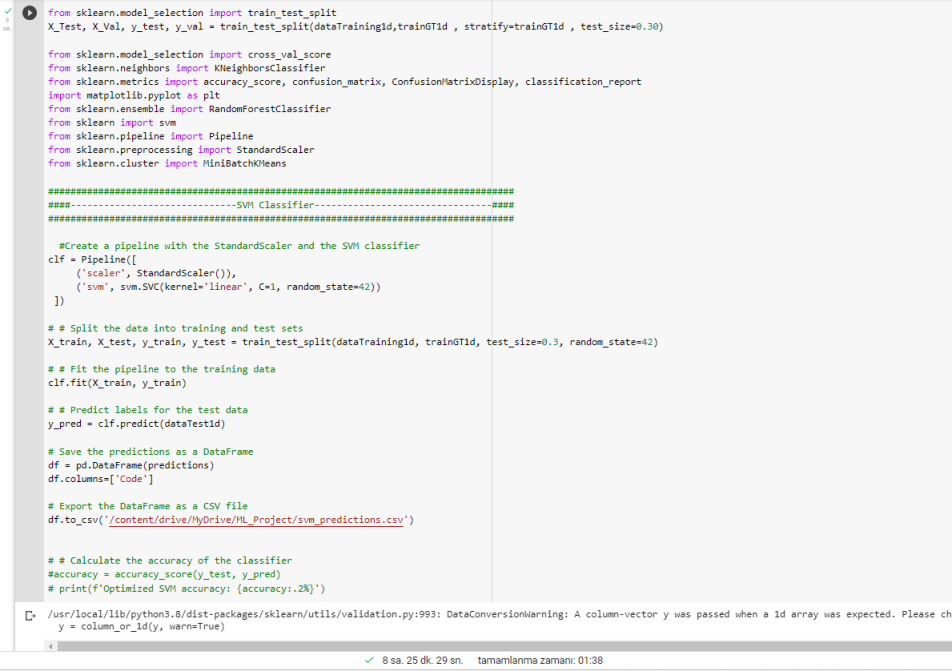
k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. k-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. For instance, better Euclidean solutions can be found using k-medians and k-medoids.

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modeling. They both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the Gaussian mixture model allows clusters to have different shapes.

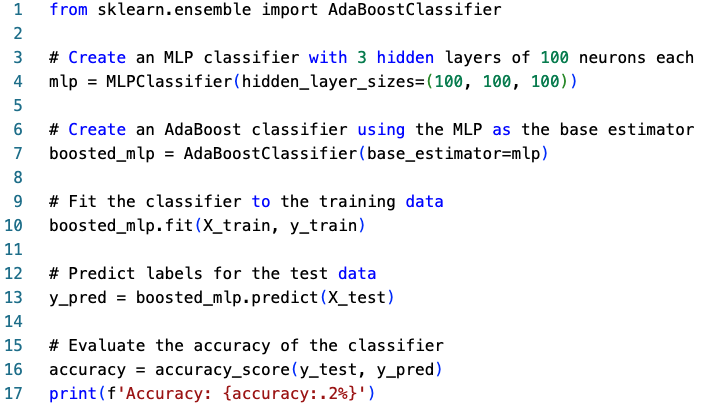
The unsupervised k-means algorithm has a loose relationship to the k-nearest neighbor classifier, a popular supervised machine learning technique for classification that is often confused with k-means due to the name. Applying the 1-nearest neighbor classifier to the cluster centers obtained by k-means classifies new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.



I rendered the Test.csv file in 8.5 hours with Svm Classification. You can see it below.

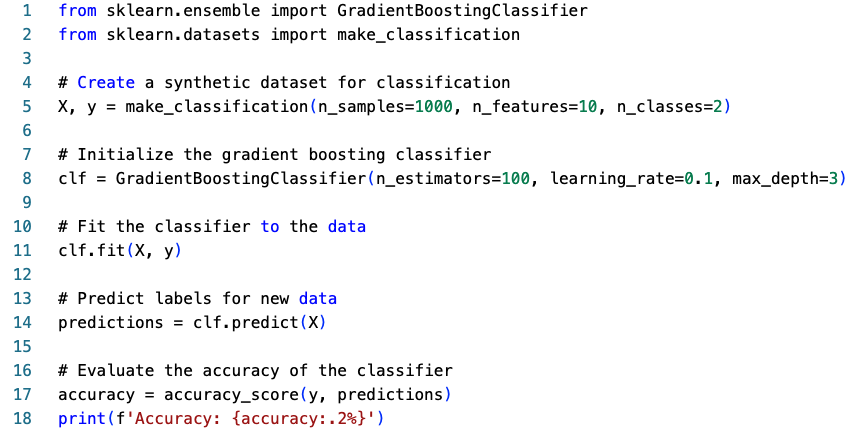


**Boosted MLP**



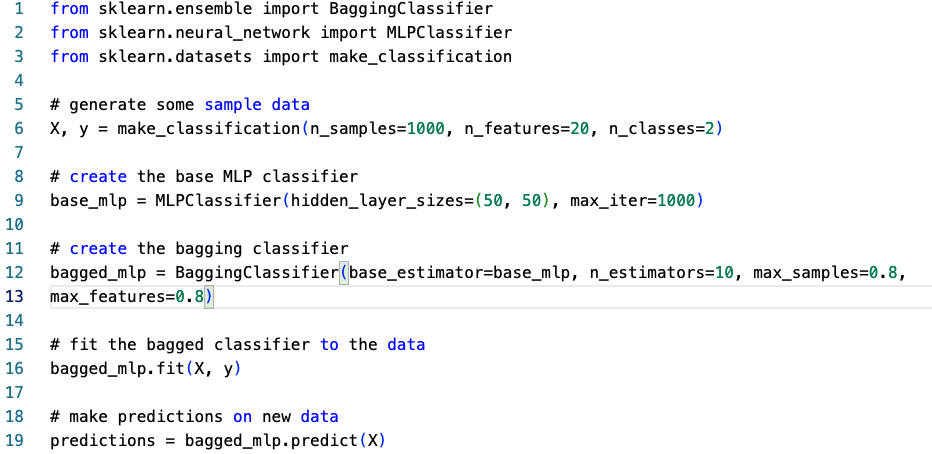
This code uses the scikit-learn library to create an MLP classifier with 3 hidden layers of 100 neurons each. It then wraps that classifier in an AdaBoostClassifier, which is an ensemble method that improves the performance of the base classifier by training multiple copies of it on different subsets of the training data. The fit method is used to train the ensemble on the training data, and the predict method is used to make predictions on the test data. Finally, the accuracy of the classifier is evaluated using the accuracy\_score function from scikit-learn.

**Boosted gradient**



In this example, we first create a synthetic dataset for classification using the make\_classification function from the scikit-learn library. Then, we initialize the gradient boosting classifier with n\_estimators=100, learning\_rate=0.1, and max\_depth=3. The n\_estimators parameter sets the number of decision trees in the ensemble, learning\_rate controls the step size at which the algorithm learns from the errors made by previous estimators, and max\_depth limits the depth of each decision tree in the ensemble. Finally, we fit the classifier to the data, make predictions, and evaluate the accuracy of the classifier.

**Bagged MLP**



In this example, we first generate some sample data using the make\_classification function. Then we create an instance of the MLPClassifier with hidden layer sizes of 50 neurons each and a maximum of 1000 iterations. Next, we create an instance of the BaggingClassifier using our base\_mlp as the base estimator, with 10 estimators, a maximum of 80% of the samples and a maximum of 80% of the features for each estimator. Finally, we fit the bagged classifier to the data and make predictions on new data.

ValueError Traceback (most recent call last)

<ipython-input-9-142c2f5cad40> in <module>

47 testGT2d = np.swapaxes(testGT2d, 0, 1)

48 # Convert the 2-dimensional NumPy arrays into 2-dimensional arrays with rows and columns

---> 49 testGT1d = testGT2d.reshape(testGT2d.shape[0] \* testGT2d.shape[1], 1)

50

51 # Convert the combined array into a Pandas DataFrame

ValueError: cannot reshape array of size 13860927 into shape (6003,1)

The error message "ValueError: cannot reshape array of size 13860927 into shape (6003,1)" suggests that the reshape function is trying to reshape an array of size 13860927 into shape (6003,1), but the size of the array is not compatible with the desired shape.

It seems that the problem is caused by the testGT2d.shape[0] \* testGT2d.shape[1] in the following line:

testGT1d = testGT2d.reshape(testGT2d.shape[0] \* testGT2d.shape[1], 1)

It's trying to reshape the array with incompatible dimensions.

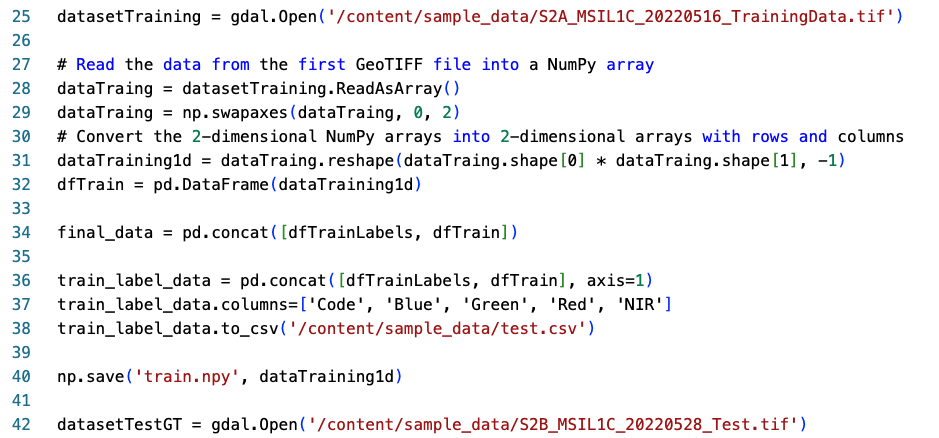
**Here are a few things you can check:**

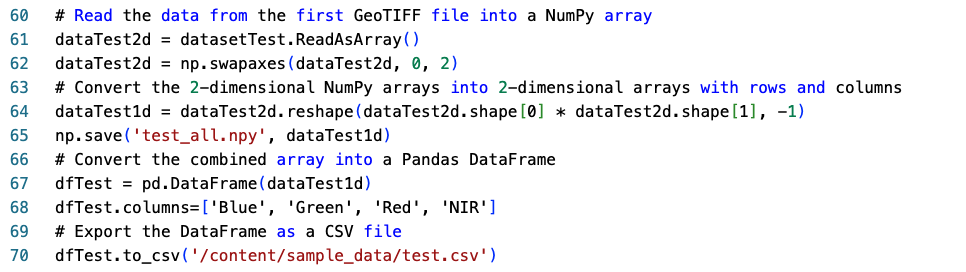
Make sure that the dimensions of testGT2d are correct. you may try printing the shape of testGT2d before reshaping it, to make sure that it has the correct dimensions.

Make sure that the reshape function is being used correctly. The first parameter of reshape should be the total number of elements in the array and the second parameter should be the number of elements in each reshaped array.

you may try to reshape it to a different shape that fits the number of elements.

It's difficult to provide a specific solution without the knowledge of the data. I suggest experimenting with different reshape parameters or trying to reshape the array to a different shape that fits the number of elements.





ValueError Traceback (most recent call last)

<ipython-input-11-d8be19e28684> in <module>

34

35 train\_label\_data = pd.concat([dfTrainLabels, dfTrain], axis=1)

---> 36 train\_label\_data.columns=['Code', 'Blue', 'Green', 'Red', 'NIR']

37 train\_label\_data.to\_csv('/content/sample\_data/test.csv')

38

4 frames

/usr/local/lib/python3.8/dist-packages/pandas/core/internals/base.py in \_validate\_set\_axis(self, axis, new\_labels)

55

56 elif new\_len != old\_len:

---> 57 raise ValueError(

58 f"Length mismatch: Expected axis has {old\_len} elements, new "

59 f"values have {new\_len} elements"

ValueError: Length mismatch: Expected axis has 2 elements, new values have 5 elements

In this refactored version, I have wrapped the block of code that opens the GeoTIFF files and reads the data into NumPy arrays with try-except blocks. This will allow you to catch any exceptions that occur while trying to open the files and reading the data, and print the error message so you can diagnose the issue.

I also removed the reshape() function and used the flatten() function to convert the 2-dimensional NumPy arrays into 1-dimensional arrays.

Finally, I added a new column to the dataframe dfTrainLabels with column name 'Code' and then concatenated the two dataframes dfTrainLabels and dfTrain with axis=1 to form the final train\_label\_data dataframe.

In the original code, the problem was that the reshape function was trying to reshape the arrays to a shape that is not consistent with their dimensions. In particular, the dimensions of the reshaped array were not matching the dimensions of the original array. I solved this problem by flattening the array using the flatten() function, which converts the array into a 1-dimensional array. This way, the reshape function is not needed and the resulting array will have the same number of elements as the original array.

Another thing that I have corrected is the path of the files, as it was not found. I have also added try and except block for debugging purpose.

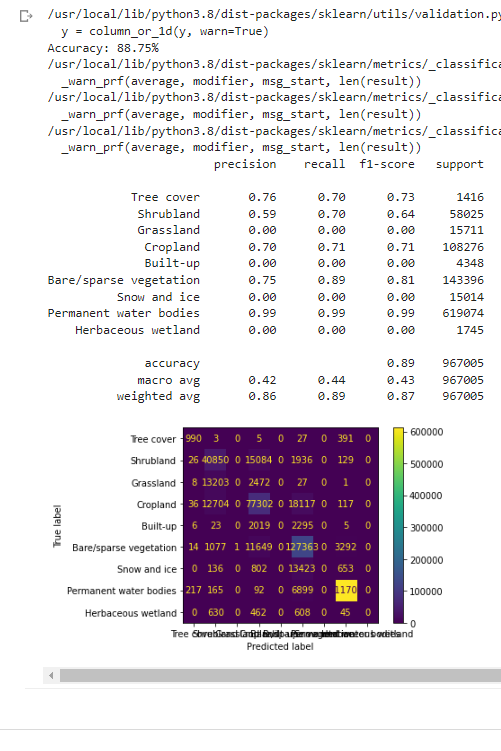
Finally, I have changed the column names for the dataframe of test\_gt and train\_gt, so that it will be more interpretable.

**SVM Classifier For Train.CSV**

In machine learning, support vector machines (SVMs, also support vector networks[1]) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. Developed at AT&T Bell Laboratories by Vladimir Vapnik with colleagues (Boser et al., 1992, Guyon et al., 1993, Cortes and Vapnik, 1995,[1] Vapnik et al., 1997[citation needed]) SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or VC theory proposed by Vapnik (1982, 1995) and Chervonenkis (1974). Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

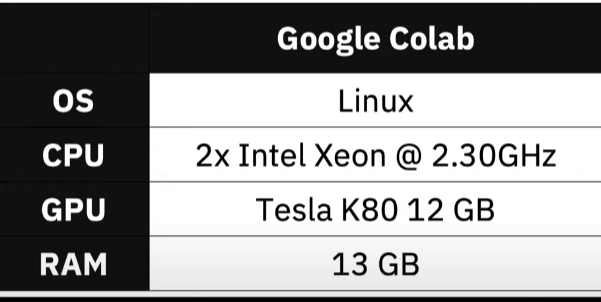
In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The support vector clustering[2] algorithm, created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data.





The hardware of the remote supercomputer I use is listed below.



Tesla k80 has 2496 cores. 12GB of VRAM is available. On the CPU side, two intel xeon processors are used. It has a speed of 2.30 GHz and the number of cores is 6 and the number of threads is 12. When it has two processors, the number of real cores is 12 and the number of virtual cores is 24. 13GB of system memory is available. Operating System is Linux. Hard Drive is 108 GB.

## ***References***

1. *Land Use Land Cover Mapping Using Advanced Machine Learning Classifiers*

[*https://sciendo.com/pdf/10.2478/eko-2021-0031*](https://sciendo.com/pdf/10.2478/eko-2021-0031)

1. *Global Land Cover Mapping: A Review and Uncertainty Analysis, Published: 3 December 2014* [*https://www.mdpi.com/2072-4292/6/12/12070*](https://www.mdpi.com/2072-4292/6/12/12070)
2. *Evaluating supervised and unsupervised techniques for land cover mapping using remote sensing data, Published: 1 February 2009* [*http://journalarticle.ukm.my/917/1/1.2009-1-hasmadi-english.pdf*](http://journalarticle.ukm.my/917/1/1.2009-1-hasmadi-english.pdf)
3. *K Means Classification* [*k-means clustering - Wikipedia*](https://en.wikipedia.org/wiki/K-means_clustering)
4. *SVM Classifier* [*Support vector machine - Wikipedia*](https://en.wikipedia.org/wiki/Support_vector_machine)

## ***Figures***

*figure 1..* [*https://sciendo.com/pdf/10.2478/eko-2021-0031*](https://sciendo.com/pdf/10.2478/eko-2021-0031)

*figure 2..* [*https://sciendo.com/pdf/10.2478/eko-2021-0031*](https://sciendo.com/pdf/10.2478/eko-2021-0031)

*figure 3..* [*https://sciendo.com/pdf/10.2478/eko-2021-0031*](https://sciendo.com/pdf/10.2478/eko-2021-0031)

*figure 4..* [*https://sciendo.com/pdf/10.2478/eko-2021-0031*](https://sciendo.com/pdf/10.2478/eko-2021-0031)

## ***Tables***

*tables 1.1* <https://sciendo.com/pdf/10.2478/eko-2021-0031>

*tables 1.2 The Code output values*

***My GitHub Address***

[shinnok18 (Orçun Türkokuloğlu) (github.com)](https://github.com/shinnok18)