Image Segmentation For Environmental Changes

Vichaka Houi and Jimmie Cox

Department of Computer Science and Engineering

Wright State University, Dayton, OH  
Email: [houi.3@wright.edu](mailto:houi.3@wright.edu) and cox.366@wright.edu

**Abstract.**

This study addresses the challenge of classifying terrain images and its type from satellite imagery whilst using machine learning techniques. Accurate terrain classification is crucial for applications such as monitoring, city development and urban development. The research used “Earth Terrain, Height, and Segmentation Map Images” by Thomas Pappas comprising of 15,000 images divided into to each of the following: terrain, height and masking image (5000 each). Using feature extraction, RGB values were extracted from the terrain and height features from the height image. Due to computational restraints, the dataset was decreased to 3,000 and images were reduced from 512 x 512 to 64 x 64 pixels. Three different training models were used: Support Vector Machine (SVM), Random Forest, K-nearest neighbors (KNN). The SVM model had achieved an overall accuracy of 94%, with micro precision being at 91%, micro f1 at 93% and micro recall at 93%. It also had macro evaluations at 91% for precision, 94% for recall, and 92% for f1. While SVM performz well, when compared to models like KNN and random forest. Both models can work and achieve near perfect scores in all evaluation metrics. Although KNN is slower, but better for smaller datasets. With random forest, tuning is necessary to achieve such results.

**Keywords:** Support Vector Machine, Random Forest, K-Nearest Neighbors

1. Introduction

In the realm of computer vision, simply classifying an entire image with a single label often does not give a comprehensive understanding of what the image contains. Most people are often interested in what individual parts make up an image like specific areas in the background or foreground. This is where the idea of image segmentation becomes essential for a machine to better understand an image and allow for things like pattern recognition and image understanding [1]. By dividing an image into meaningful regions and classifying each region, image segmentation allows a machine to better interpret and analyze visual data much more effectively making it an important part of this image understanding [2]. This mimics how humans can actually perceive and understand the individual components of an image. This use of segmenting an image into various regions can be used in a wide variety of fields like traffic control systems, facial recognition, fingerprint recognition, medical image detection, farming, and forensic analysis [2] [1].

But how might we use image segmentation to analyze changes in the earth’s environment? By segmenting an image into distinct regions, various features—such as color, grayscale intensity, or spatial texture—can classify individual pixels into their different region classes [2]. For terrain segmentation analysis, the use of terrain and height images are utilized in color and grayscale respectively. This use of terrain and height images are used to produce segmentation issues that split the pixels into different regions that represent different terrains. This approach provides a fundamental solution for visually tracking and understanding changes in the terrain environment over time.

1. Dataset

The dataset we are using is sourced from Kaggle under the title “Earth Terrain, Height, and Segmentation Map Images” by Thomas Pappas. It originally contains 15,000 images, with 5,000 images each for terrain, height, and segmentation. To improve computational efficiency, we reduced the dataset to 3,000 images, with 1,000 images of each type (terrain, segmentation, and height) [3]. These images are 512 x 512 and include important features which include RGB values in three channels and height image values ranging from 0 to 255. A value of 0 represents sea level, while 255 indicates the highest points, such as mountains [3]. For the region classes, the dataset defines them as water, grassland, forest, hills, desert, mountain, or tundra. These classes are defined with their corresponding RGB values which are as follows: (17, 141, 215), (225, 227, 155), (127, 173, 123), (185, 122, 87), (230, 200, 181), (150, 150, 150), and (193, 190, 175) [3].

Preprocessing the data is also another crucial step in preparing the dataset for model training. The primary preprocessing step involved resizing the images to improve computational efficiency. To achieve this, we resized the original images from 512 x 512 pixels to 64 x 64 pixels. The next step was to group the three types of images (height, segmentation, and terrain) together based on the image numbers at the start of their filenames. For example, the image "0001\_h.png" was grouped with its corresponding "0001\_t.png" and "0001\_i2.png". After reducing the dataset to 3,000 images, this method allowed us to form 1,000 groups, each containing one image of each type. These groups can be treated as individual samples.

Another key step of the preprocessing step which was mentioned above is the feature extraction from the images. For the feature extraction, we collected three channels (RGB) from the terrain images and height values from the height images, ranging from 0 to 255. These extracted features come from each individual pixel represented in the terrain and height images. This gives us four features to work with when training the three models as follows: Support Vector Machine, Random Forest, and K Nearest Neighbors. The height values represent terrain elevation, with 0 indicating sea level and 255 representing the peak of a mountain. These values are especially useful for distinguishing between similar colors in the segmentation classes.

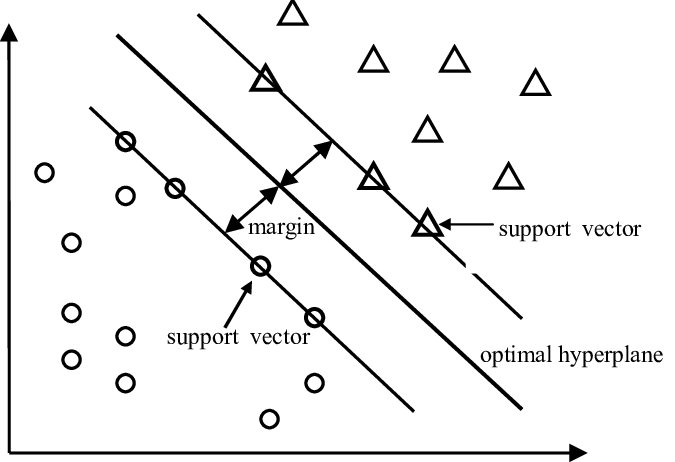
Then the actual splitting of the dataset is done in an 80/20 split where 80% of the 1000 grouped images were used for training and 20% were used for testing. This translates to 800 groups for training and 200 groups for testing. We use the entire image as a sample rather than taking the individual pixels as samples, so we look at the image as a whole.

1. Methodology

This section offers our three models and their respective methodologies. Subsection 3.1 covers the Support Vector Machine (SVM), providing a general overview of its functionality and its application to image segmentation in general. Similarly, subsections 3.2 and 3.3 discuss the methodologies for Random Forest and K-Nearest Neighbors (KNN), respectively, explaining their general use cases and their specific roles in segmentation tasks.

**3.1 Support Vector Machine**

Support Vector Machine (SVM) is a machine learning model that in essence searches for the best hyperplane that will split datapoints up into two regions or classes. That being said, it can be broken down as a classifier that is great for solving practical binary classification problems [4]. This means that the original SVM that was built was only built for binary classification. The general idea is that the model is trying to maximize and find the largest interval defined in the feature space [2]. This being the distance between the optimal hyperplane and its corresponding support vectors. These support vectors are the closest data points from each side of the hyperplane otherwise known as the two classes [2] [5]. This general representation of an SVM can be seen in the representation below (see Fig. 1).



**Fig. 1** A basic SVM model that separates two classes through the use of an optimal hyperplane. This figure demonstrates the general structure of the SVM model.

But since we are working on multiple classes and a non-linear relationship between datapoints, we will need to effectively change the way in which the SVM operates. This is why the SVM also supports the use of kernel techniques, so that it can be used on non-linear relationship datapoints [2].

This kernel parameter that we use is known as the Radial Basis Function (RBF) or known as the Gaussian Kernel [5]. The basic idea behind a non-linear SVM is that it tries to transform the input vectors into a higher dimensional feature space. So, while the hyperplane, the line that separates the classes, is no longer linear, it will get transformed into a higher dimensional feature space. The RBF is the most popular and most widely used kernel for different applications [5]. The basic understanding of how a non-linear hyperplane would look like as well as how the kernel affects it into a higher dimensional space can be seen below (Fig. 2).

A diagram of a diagram of a diagram

Description automatically generated

**Fig. 2** SVM Non-Linear Hyperplane being transformed into a higher dimensional plane to help better separate datapoints.

After using the Radial Basis function for the kernel, there are two other parameters that were crucial in our SVM model. These two parameters were the C and max iteration parameters. The C parameter is used in Support Vector Machines as the regularization parameter [5]. This regularization parameter is useful to help prevent overfitting. The primary goal of setting a lower C value produces a hyperplane with a larger margin separation between it and the support vectors [5]. This allows the model to potentially misclassify some points during training, but will produce a better model for generalizing to new images and their pixels. On the other hand, a high C value will emphasize minimizing the training error which will then give us a worse generalization [5]. This is ultimately a tradeoff between having either a high or low bias or variance depending on the problem we’re using our model for. In our case, we used the default parameter setting for C as 1.0 as regularization of the model was not really required. The max iterations are another important parameter that we used for the SVM. If left unset, the hyperplane after each iteration will continue to look for the most optimized hyperplane, but in our case, this was taking too long. We capped the number of iterations at 1000 as that still produced a good accuracy without taking a long time. This was an important tradeoff that we did in terms of computational time and how good our accuracy was.

Finally, in order to deal with the SVM being used for binary classification, we need to use a method called one versus one to build our model for multiple class classification. This one versus one method constructs multiple hyperplanes where each hyperplane is trained with two-class data sets [4]. This means that based on the number of classes you have *k*, the formula *k(k-1) / 2* will determine the number of hyperplanes or binary classifications you have to compare. After doing the comparison between datapoints using hyperplanes, a voting system is used to correctly classify datapoints to a specific class [4].

**3.2 Random Forest**

Random Forest is a machine learning model that is a type of ensemble learning. The model itself builds on multiple decision trees during training and combines their predictions to make a final classification. Each tree in the forest is trained on a random subset of the data which introduces diversity and reduces overfitting. By averaging the outputs of multiple trees, random forest achieves high accuracy and robustness to noise.

Within this research, RandomForestClassifier from Scikit-learn was utilized. 100 trees were also employed with n\_estimators=100. The depth of the trees, the max\_depth was not defined and thus, allowing the model to freely and adaptively determine the data by getting it to the pure level. This method allows us to best and accurately classify each pixel. Although this may result in overfitting.

To avoid complete overfitting, setting up depth levels will help the model find balance between overfitting and under-fitting. In this research, we will define max\_depth from 1 to 20. Although in this report we demonstrate 3, 8, 10, and 20. Having the low numbers will help as it tells areas or specific depth levels where it may be under-fitting. By increasing depth level, we can home to search for a good generalization while also trying to stay away from overfitting.

**3.3 K-Nearest Neighbors**

K-Nearest Neighbors (KNN) is a classification model that makes predictions by finding the “k” nearest data point to its specific center point of start. It predicts the class of a data point by identifying its “k” nearest neighbors in a created or developed feature space. The prediction is based on the majority class among these neighbors that determine the prediction.

Within this research, KNeighborsClassifier implementation from Scikit-learn library was utilized. The values of *k* featured in this research valued from 1 to 14 to identify different and optimal settings of *k*. The reason for these k values was to help find a balance between overfitting and underfitting. Although in this report, we focus on k=5 as it showed the highest results amongst them all. To ensure no biased from the dataset, all the features extracted were normalized to help ensure the differences in scale did not bias the distance calculations central to KNN. In this model, a specific weight was also used with it being the ‘distance’ weight type. Weight was chosen for handling the specific traits of features the pixels had. RGB class values that are closer to the specific RGB pixel to predict would help better identify its own class. The model was evaluated for its performance in terms of accurately predicting terrain types based on pixel level features. It used performance metrics such as accuracy, precision, recall, F1 score micro, F1 score macro, and confusion matrix.

1. Experimental Results

**4.1 Support Vector Machine**

The results from the SVM model were evaluated using Scikit-learn’s evaluation metrics. These included using a classification report, individual class accuracy, individual precision, individual recall, confusion matrix, overall accuracy score, overall precision score, overall recall score, and overall F1 macro and F1 micro score.

The classification report was able to provide the accuracy, precision, recall, and F1-score for each class. Table 1 highlights some interesting results, specifically in the performance of the water and mountain class. For the remaining five classes, their metrics varied, but they generally fell within the range of 75% to 100%.

The results for the different individual classes in Scikit-learn’s evaluations are below in Table 1, and the bar graph results can be seen below (see Fig. 1).

**Table 1.** Table that shows the evaluation metrics for the SVM model that holds the accuracy, precision, recall, and F1 score of each class.

A table with numbers and text

Description automatically generated

A graph of different colored lines

Description automatically generated

**Fig 1.** Figure that shows the evaluation metrics for the SVM model that holds the accuracy, precision, recall, and F1 score of each class in a bar graph.

The confusion matrix also reveals some insights as they align with the results from the classification report defined by Scikit-learn. We can see that a majority of the classified pixels are correctly classified as they fall along from the top left corner to the bottom right, indicating accurate predictions for a majority of the classes.

Our confusion matrix can be seen in Table 2 below.

**Table 2.** Table that shows our confusion matrix between differing classes.

A screenshot of a graph

Description automatically generated

Our final evaluation metric focused on seeing the overall scores in regards to precision, recall, micro F1 score, and macro F1 score.

This information can be seen below in Table 3 below.

**Table 3.** Table that shows our overall metrics provided by sklearn.

A blue rectangle with black text

Description automatically generated

**4.2 Random Forest (no depth defined)**

The results from Random Forest were evaluated the same as the SVM model to bring equal comparison.

**Table 4.** Table that shows the evaluation metrics for the Random Forest model that holds the accuracy, precision, recall, and F1 score of each class.

A screenshot of a table

Description automatically generated

**Fig 2.** Figure that shows the evaluation metrics for the Random Forest model that holds the accuracy, precision, recall, and F1 score of each class.

In table 5, the confusion matrix shows us that after the model reaches the furthest depth level, at high levels of accuracy it can classify each pixel. There are indeed incorrect predictions within the matrix prediction, but it is still able to correctly identify a high majority of the specific class.

**Table 5.** Table that shows our confusion matrix between differing classes.

A table with numbers and a number

Description automatically generated

**Table 6.** Table that shows our overall metrics provided by Scikit-learn.

**A blue and white box with black text

Description automatically generated**

**4.3 Random Forest (depth = 3)**

**Table 6.** Table that shows the evaluation metrics for the random forest model that holds the accuracy, precision, recall, and F1 score of each class.

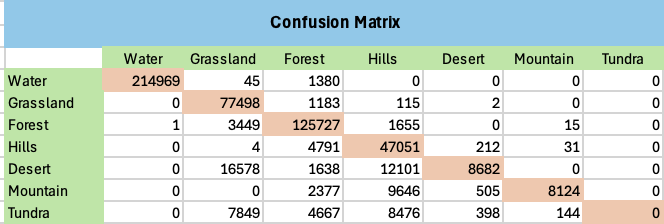
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Random Forest** | | | | |
|  | Accuracy | Precision | Recall | F1-score |
| Water | 0.99 | 1.00 | 0.99 | 1.00 |
| Grassland | 0.98 | 0.74 | 0.98 | 0.84 |
| Forest | 0.96 | 0.89 | 0.96 | 0.92 |
| Hills | 0.90 | 0.60 | 0.90 | 0.72 |
| Desert | 0.22 | 0.89 | 0.22 | 0.36 |
| Mountain | 0.39 | 0.98 | 0.39 | 0.56 |
| Tundra | 0.00 | 0.00 | 0.00 | 0.00 |

In figure 3, when compared to the previous version with no defined depth level, we notice almost complete prediction failure for Tundra. Other classes of terrain had also been heavily impacted. Most likely due to depth level underfitting on some of the pixels.

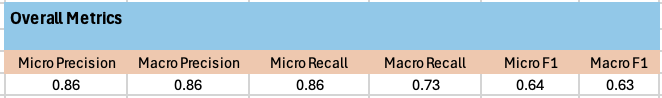
**Fig 3.** Figure that shows the evaluation metrics for the Random Forest model that holds the accuracy, precision, recall, and F1 score of each class.

In table 7, we can see further on what the model is predicting for tundra. For all classes of terrain we can see that water is still predicted highly well and correctly. Other classes tend to struggle too like desert, mountain and especially tundra.

**Table 7.** Table that shows our confusion matrix between differing classes.

****

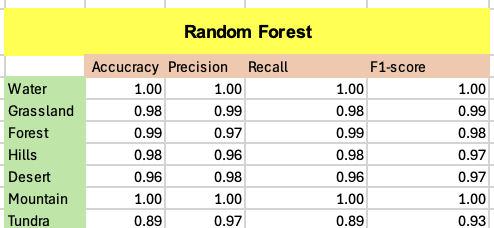
**Table 8.** Table that shows our overall metrics provided by Scikit-learn..

****

**4.4 Random Forest (depth = 8)**

In table 9, we notice that by increasing depth level, each performance metrics increase a lot more. Further enhancing the specific details towards each type of terrain.

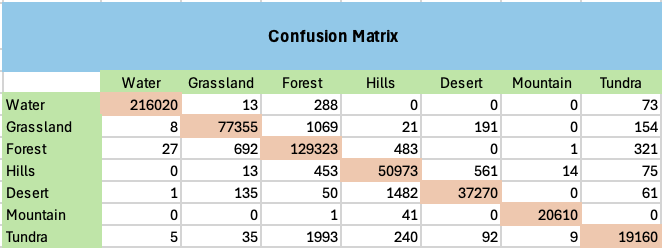
**Table 9.** Table that shows the evaluation metrics for the random forest model that holds the accuracy, precision, recall, and F1 score of each class.

****

In figure 4, it shows further information about accuracy, precision, recall and f1-score. It is much higher compared to the one with depth level = 3. Tundra is still the lacking class compared to the others. Although now for mountain, it had been able to achieve relatively high performance like water terrain did.

**Fig 4.** Figure that shows the evaluation metrics for the Random Forest model that holds the accuracy, precision, recall, and F1 score of each class

**Table 10.** Table that shows our confusion matrix between differing classes.

****

**Table 11.** Table that shows our overall metrics provided by Scikit-learn.

**A blue rectangle with black text

Description automatically generated**

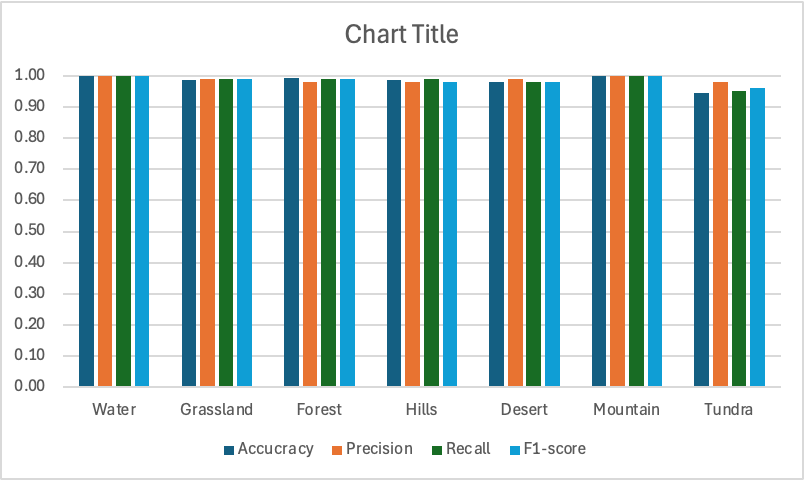
**4.5 Random Forest (depth = 10)**

When working with a slightly further depth level, we get closer to perfect performance for each of the evaluation metrics. Although classes like tundra could still use improvement in identifying more correct classes.

**Table 12.** Table that shows the evaluation metrics for the random forest model that holds the accuracy, precision, recall, and F1 score of each class.

**A screenshot of a table

Description automatically generated**

**Fig 5.** Figure that shows the evaluation metrics for the Random Forest model that holds the accuracy, precision, recall, and F1 score of each class

**Table 13.** Table that shows our confusion matrix between differing classes

A table with numbers and a number

Description automatically generated

**Table 14.** Table that shows our overall metrics provided by Scikit-learn.

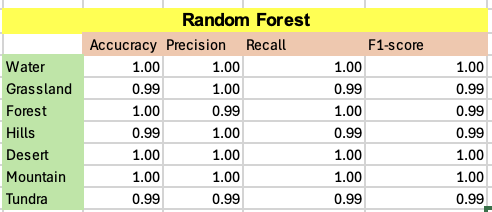
A blue rectangle with black text

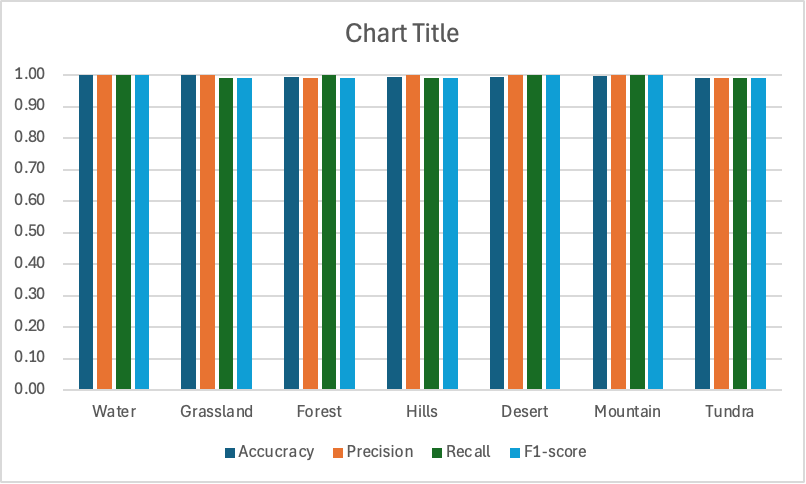
Description automatically generated

**4.6 Random Forest (depth = 20)**

Finally, we now try with a depth level of 20. By increasing the max depth level we hope to further get higher results than one with 10. In table 15, we can now see that each evaluation metric has almost achieved near or close to 100%.

**Table 15.** Table that shows the evaluation metrics for the random forest model that holds the accuracy, precision, recall, and F1 score of each class.

****



**Fig 6.** Figure that shows the evaluation metrics for the KNN model that holds the accuracy, precision, recall, and F1 score of each class

**Table 16.** Table that shows our confusion matrix between differing classes

**A table with numbers and a number

Description automatically generated**

**Table 17.** Table that shows our overall metrics provided by Scikit-learn.

**A blue rectangle with black text

Description automatically generated**

**4.7 K-Nearest Neighbors**

The results from KNN were evaluated the same as the SVM model to bring equal comparison.

**Table 18.** Table that shows the evaluation metrics for the KNN model that holds the accuracy, precision, recall, and F1 score of each class.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **KNN** | | | | |
|  | Accuracy | Precision | Recall | F1-score |
| Water | 1.00 | 1.00 | 1.00 | 1.00 |
| Grassland | 0.99 | 0.99 | 0.99 | 0.99 |
| Forest | 0.99 | 0.99 | 0.99 | 0.99 |
| Hills | 1.00 | 0.99 | 0.99 | 0.99 |
| Desert | 1.00 | 1.00 | 1.00 | 1.00 |
| Mountain | 1.00 | 1.00 | 1.00 | 1.00 |
| Tundra | 0.99 | 0.99 | 0.99 | 0.99 |

**Fig 3.** Figure that shows the evaluation metrics for the KNN model that holds the accuracy, precision, recall, and F1 score of each class.

**Fig 7.**

**Table 19.** Table that shows our confusion matrix between differing classes

**A table with numbers and a number

Description automatically generated**

**Table 20.** Table that shows our overall metrics provided by sklearn.

A blue and black text on a white background

Description automatically generated

1. Discussion

**5.1 Support Vector Machine**

For the SVM, there are plenty of insights that can be taken from the classification results. Table 1, the classification report, highlights the evaluation metrics behind the class accuracy, precision, recall, and F1-score. Notably, the water and mountain classes achieved the highest metrics possible in all four of these metrics. This is because of the influence of the heigh feature that we pulled from the height images. As stated in the dataset, the height value ranges from the value 0 (sea level or very low terrain indicating water) to 255 (high elevations such as a mountain). When these datapoints got mapped into a higher dimensional space, they help in terms of being able to distinguish these datapoints from others, making them easier to separate with a hyperplane. For the other five classes, the variation in these metrics show more variability. That’s because, unlike the height value with water and mountain, we have more variability in height between classes that makes the separation of these datapoints less straightforward.

The confusion matrix helps to provide a visual understanding of these same metrics by plotting out the actual pixel class label and the predicted class label by the model. Table 3 displays our confusion matrix, and the classes have a majority of pixels labeled correctly. This indicates that each class has at least 50% accuracy for correctly predicting the correct pixels. These findings align correctly with our classification report. From Scikit-learn, we were able to find the overall accuracy from average of individual accuracies of classes which was at 93% giving us a decent model for predicting terrain image segmentations.

Finally, we evaluated the overall metrics to see the micro and macro precision, recall, and F1-score. These ranged between 91% and 94%. With an overall accuracy of 93%, these metrics remain consistent, indicating a fairly well-performing model. These results indicate that the model is not overfitting nor underfitting. This helps to support the conclusion that this model can effectively predict images of terrain segmentation.

**5.2 Random Forest**

For Random Forest, we tested several different types of depth levels to test the performance of each one. Searching for ideal depth levels and compare how they fare to each other as well. To remind once again, we had used no depth level, max\_depth=3, max\_depth=8, max\_depth=10, and max\_depth=20.

**5.2.1 Random Forest (No max depth)**

When working with no defined depth level, Random Forest is able to continuously make decision trees to further define the perfect classification for each pixel. But may in the end result in overfitting of the model.

Firstly, according to table 4, we can see that the model with no depth level defined is indeed able to highly and accurately predict each terrain type. Figure 2 further enhances that vision of claim. This suggests the model may very well be overfitting potentially.

Lastly, we have the confusion matrix. This is insightful as it provides further clarification that this model is not a complete 100% in all of its evaluation metrics. For each class, we notice that they make small number of incorrect predictions in proportion to its overall correct predictions. Or a very high precision rate. Table 6 helps prove that claim and potentially further suggest that the model may be overfitting.

**5.2.2 Random Forest (max\_depth = 3)**

For starters, we chose a max\_depth level of 3 as it is highly likely to underfit its training samples and predictions. This helps us understanding the struggles for classification the model may have early on.

Firstly, in reference to table 6, we can see varying accuracies, precision, recall and F1-score. Water stands to perform well most likely due to its RGB pixel values are much clearer to define than the others. It also has its own unique height to help it out. Grassland, forest, hills and desert also have high accuracy rate. But that high accuracy may be due to them predicting other classes as themselves. It shows a bit of decrease in performance compared to the others. Although its recall meaning it able is able to reasonably predict the right classes that belong to it. The other terrains like hills, desert, mountain and tundra may be experiencing a class imbalance as well. Their accuracy is relatively low, but is highly likely able to predict itself correctly. Although this assumption is when the model is highly confident due to high precision and low recall. It also appears that the model with this depth level is also unable to identify any tundra classes. Refer to figure 3 for a graphical representation of this.

We also deploy a confusion matrix to help identify how the model is predicting. In reference to table 7, we can see that water is highly accurate. It occasionally predicts forest and some grassland but we can for sure say that most of the time it will predict water when the pixel is water. And also, no other terrain really gets classified as water. We also notice that grassland is relatively similar to water in performance although its precision may not be the best in terms of proportions. As we further go down the list of terrains with less and less samples, we notice performance is also getting lower. Especially the class tundra, it is predicting a lot of grassland, forest and hills. Table 8 also helps showcase this claim.

**5.2.3 Random Forest (max\_depth = 8)**

Next, we try to increase the depth level to better help fit the model in training. Using max\_depth levels with 7 or 9 produced similar results or are close towards 8.

In reference to table 9, we notice the evaluation metrics here are much better compared to the previous max depth level used. The performance for water remained pretty much the same. Grassland had also come close to such performance followed by forests, hills, and desert. Although surprisingly, mountain performed just as good as water. It may most likely result in its height feature that best help distinct itself. Tundra was also able to big improvements with an accuracy of 89%. This suggest that it has a decently strong performance especially compared to the previous one. It is also able to accurately classify itself pretty well. But its recall is not as good though as more samples of this may be needed. Figure 4 helps showcase this claim.

In reference to table 10, it helps understand all the predictions being made. We notice more incorrect predictions with water for other terrains. Although it is only a small amount in proportions to their actual predictions of themselves. Although proportionally, tundra is still performing the worst amongst all of them. A good portion of it is being incorrectly identified as forest.

Lastly, in terms of its overall metrics, it on average performed well. Refer to table 11 for its representation. The metrics varied between 97 and 98%. Although it may also mean that the model is starting to over fit.

**5.2.4 Random Forest (max\_depth = 10)**

Next up we tried a depth level of 10. The model itself was able to improve further from depth level 8. In table 12, we still notice that water and mountain have basically the same performance as it did with depth level 8. Although compared to before, we have notice more improvements in forest, hills, desert and especially tundra. Figure 5 helps us give that visualization to compare with depth level 8.

We also again use a confusion matrix to help identify the predictions made. As compared to previously depth, we notice slight improvements in all the terrain classes. But, tundra was able to make some noticeable improvement across its performance metrics. Resulting in higher overall metrics for the mode in table 14.

**5.2.5 Random Forest (max\_depth = 20)**

Lastly, we tested random forest with a depth level of 20. In table 16, the results were relatively similar for all terrains but with some closer to 100% across its evaluation metrics. But the model was able to show a good increase in its evaluation metrics for tundra. Figure 6 can help use see that further as well.

In the confusion matrix, we can see the amount of incorrect predictions for all terrains had decreased. We see the biggest of change and performance increase in tundra.

**5.3 K-Nearest Neighbors (KNN)**

For KNN, there were not too many things that stood out besides it’s high performance for all evaluation metrics. Accuracy, precision, recall and F1-score in table 18 were all relatively close. Figure 7 is also able to showcase that further that they are all ever so slightly worst or better than each other.

The confusion matrix for KNN was still informative in providing which predictions is the model still making wrong. The model is still not yet at 100% accuracy for all its predictions but, it is still all relative to 99% for each terrain to classify.

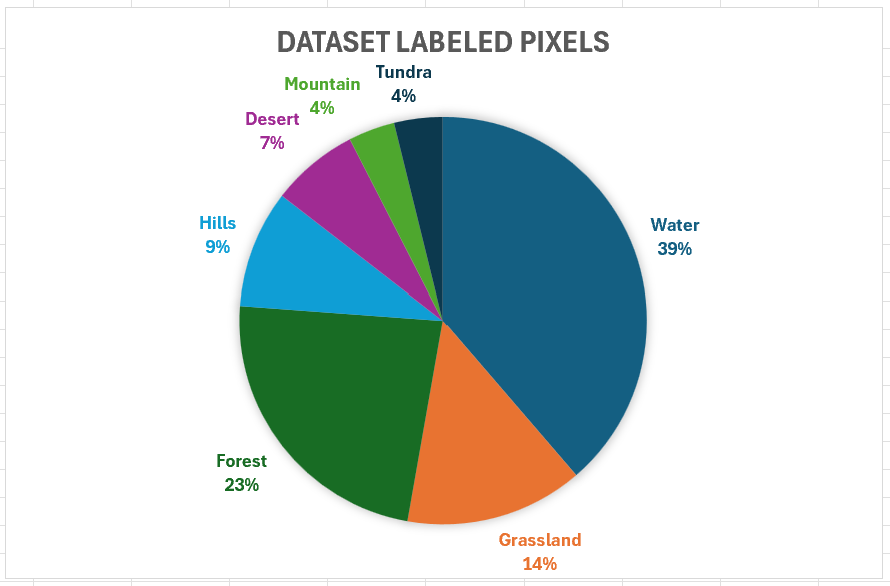
Lastly, we have the overall metrics. They all show extremely high performance. Between 99% and 100% which suggests the model is very good or overfitting.

1. Challenges and Future Steps

**6.1 Dataset Challenges**

The dataset presented two key challenges. The first was its size, consisting of 15,000 images—5,000 each of segmentation, height, and terrain. This large size of images resulted in long runtimes for our code. To address this, we reduced the dataset to 3,000 images for greater computational efficiency. The second challenge stemmed from class imbalances in the labeled training pixels. This imbalance arose due to the specific selection of images of the overall dataset in which we split to be more efficient, which impacted the representation of certain classes during training.

This challenge of an imbalance in classes can be seen below (see Fig. 8).

****

**Fig. 8** This is a pie chart showing the large imbalance between the labeled training pixels.

**6.2 Dataset Future Solutions**

For the dataset, one of the main solutions to the dataset being too large is searching for a smaller dataset or reconsidering the way in which we shrink the dataset. While reducing the dataset affects the second issue, reducing the size of the dataset isn’t inherently bad. This would be the case as long as we ensure classes stay balanced in pixel labels used for training. To ensure that the imbalance is prevented, the images can be shuffled after grouping them. The original 15,000 images can be grouped into the 5000 sets of each type, and then shuffled. Of the 5000 shuffled sets, a random approach would taken to select 1000 of these sets to ensure a more balanced and diverse set of labeled class pixels for training.

**6.3 Support Vector Machine Challenges and Solutions**

The main challenge, although not a critical one, was selecting the maximum iterations parameter. This limited the SVM from reaching the most optimal hyperplane for our datapoints. As mentioned earlier, the overall accuracy and general evaluations of our model could have been improved if we had not set this parameter at all.

A future step in ensuring the correct optimal hyperplane is reached would be to not set this parameter at all. Other methods should be looked at to try and be more computationally efficient.

**6.4 Random Forest Challenges and Solutions**

There were not many challenges for this model. Only a minor challenge was getting multiple depth levels and comparing all its evaluation metrics.

A future step to efficiently make this better is to explore for other comparison methods or libraries that can easily compare the different values.

**6.5 K-Nearest Neighbors (KNN) Challenges and Solutions**

The main challenge for KNN was validating its performance whether it was overfitting or good. The KNN model took long run times to run the model. It was not too bad to train 1 model with a specific k value but, when training to test and compare multiple k-values, it can take several hours and longer. And when testing with several different k-values, they all produced extremely high result but were all almost identical in terms of performance. This made it difficult to test and see if it can be run along different datasets of images with terrain.

A future step to ensuring the model may not be overfitting is to check different parameters for the KNN model. May also try and see if further k-values effects performance and may be able to help demonstrate generalization.

1. Conclusion

This paper addresses the problem of analyzing changes in terrain environments by applying three different models which are as follows: SVM, Random Forest, and K-Nearest Neighbors. While each model was able to produce differing results among metrics provided by Scikit-learn, the SVM performed the worst in terms of overall accuracy for creating segmentation images. Therefore, this paper highlights these differences in the metrics and the effectiveness of each model when applied to our use case of terrain image segmentation.

1. Contribution

The contributions to the presentation, report, and code were shared between both partners, Jimmie Cox and Vichaka Houi. For the entire project, Jimmie Cox focused on the SVM while Vichaka Houi focused on the Random Forest and KNN models. For the presentation, each individual contributed content related to their models they worked on. The report was ultimately a collaborative effort with each person responsible for sections about their models and working together on the remaining content. In terms of coding, Vichaka handled the preprocessing, and both worked independently on implementing their respective models.

References

1. Kaur Brar, K., Goyal, B., Dogra, A., Mustafa, M.A., Majumdar, R., Alkhayyat, A., Kukreja, V.: Image segmentation review: Theoretical background and recent advances. *Information Fusion* 114, 102608 (2025). <https://doi.org/10.1016/j.inffus.2024.102608>
2. Yang, A., Bai, Y., Liu, H., Jin, K., Xue, T., Ma, W.: Application of SVM and its Improved Model in Image Segmentation. *Mobile Networks and Applications* 27(3), 851–861 (2022). <https://doi.org/10.1007/s11036-021-01817-2>
3. Pappas, T.: Earth Terrain, Height, and Segmentation Map Images. Kaggle. <https://www.kaggle.com/datasets/tpapp157/earth-terrain-height-and-segmentation-map-images>
4. Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., Lopez, A.: A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing* 408, 189–215 (2020). <https://doi.org/10.1016/j.neucom.2019.10.118>
5. Thurnhofer-Hemsi, K., López-Rubio, E., Molina-Cabello, M.A., Najarian, K.: Earth. Published on Cornell University, <https://arxiv.org/abs/2007.08233> (2020). <https://doi.org/10.48550/arXiv.2007.08233>.