Mustererkennung und Bildverstehen Übung 2

Christian Edelmann, 3560916, st175180@stud.uni-stuttgart.de
Lars Pfeiffer, 3514519, st173192@stud.uni-stuttgart.de
Nadim Maraqten, 3384833, st160859@stud.uni-stuttgart.de
Johannes Bladt, 3541171, st175301@stud.uni-stuttgart.de
Group 1
Image tiles: 2_13, 3_14

Juni 2024

Matlab

1. **Random Forrest:** Random Forrest is a semantic segmentation method. Each decision tree is trained on a different bootstrap sample. This sample is randomly drawn from the training dataset with replacement. For each tree random features are used. Majority vote is then used to classify a segment. The correct class should be chosen, Wrong votes should be approximately equally distributed while true votes should converge on the correct class. Random Forest is a highly accurate and robust method. On the other hand a high number of trees would require a lot of computing power leading to long training and prediction times.

2. **Initialization:** The environment is cleared and prepared by closing all figures, clearing the workspace, and turning off warnings.

3. **Loading Data:** Intermediate results from the previous exercise are loaded from a .mat file. The images get resized by 0.5. For chessboard we use segment size = 15 pixels and for SLIC we use method SLIC, cluster number k = 1000 and compactness = 1.

4. **Random Forest Training:**

   (a) Different sizes of decision trees are defined: 10, 25, 50, 100, 250, 500, and 1000 trees.

   (b) For each tree size:

      i. Train a Random Forest classifier using the training features and labels.
      ii. Measure the training time.
      iii. Predict labels for the training data using out-of-bag prediction.
      iv. Construct a labeled image for the training data by segmenting the image and assigning predicted labels to each segment.
      v. Visualize the segmented training image with the predicted labels.
      vi. Save the visualization of the training image with the predicted labels.

5. **Random Forest Testing:**

   (a) Predict labels for the test data using the trained Random Forest classifier.

   (b) Construct a labeled image for the test data by segmenting the image and assigning predicted labels to each segment.

   (c) Visualize the segmented test image with the predicted labels.

   (d) Save the visualization of the test image with the predicted labels.

6. **Confusion Matrix:** Compute the confusion matrix for the test data by comparing predicted labels with ground truth labels pixel by pixel.

7. **Accuracy Calculation:**

   (a) Calculate the overall accuracy by comparing the predicted labels with the ground truth labels pixel by pixel.

   (b) Calculate the accuracy for each class by evaluating the main diagonal of the confusion matrix.

8. **Confusion Chart:** Visualize and save the confusion matrix chart with row-normalized and column-normalized values.

9. **Performance Plot:**

   (a) Plot the accuracy, precision for each class, and runtime for different tree sizes.

   (b) Save the performance plot.

Results

1) Confusion matrices using only one tree.

Confusion matrices are used to compare the accuracy of the different setups. The following table shows the 6 different classes used during the exercise.

class number	class name
1	Impervious surfaces
2	Buildings
3	Low vegetation
4	Trees
5	Cars
6	Clutter

1.1) Chessboard

As expected using only one tree the accuracy of the training data is at only 45 percent. The one tree used seems to have a very strong bias towards class 3 low vegetation as now matter the true class the majority is predicted as class 3. To ensure more robust results more than one tree is needed to reduce the risk of problems like this.

The overall accuracy of the test image is 62 percent. Buildings are the most accurate while class 5 cars only has 22 percent accuracy. Trees and low vegetation while having over 50 percent precision get mistaken for each other quite often as well.

A higher test accuracy than training accuracy, as reported here, is generally unlikely but not impossible, especially when considering a single tree in a random forest. This scenario suggests a very low model complexity, which might lead to an inability to accurately predict the potentially more complex feature landscape in the training data, resulting in lower training accuracy. However, the patterns learned by the tree might be well-represented in the test data, leading to higher test accuracy. It is shown that this situation of higher test than train accuracy is only the case for 1 tree, supporting the often observed phenomenon of slight overfitting for the other n_{trees}.

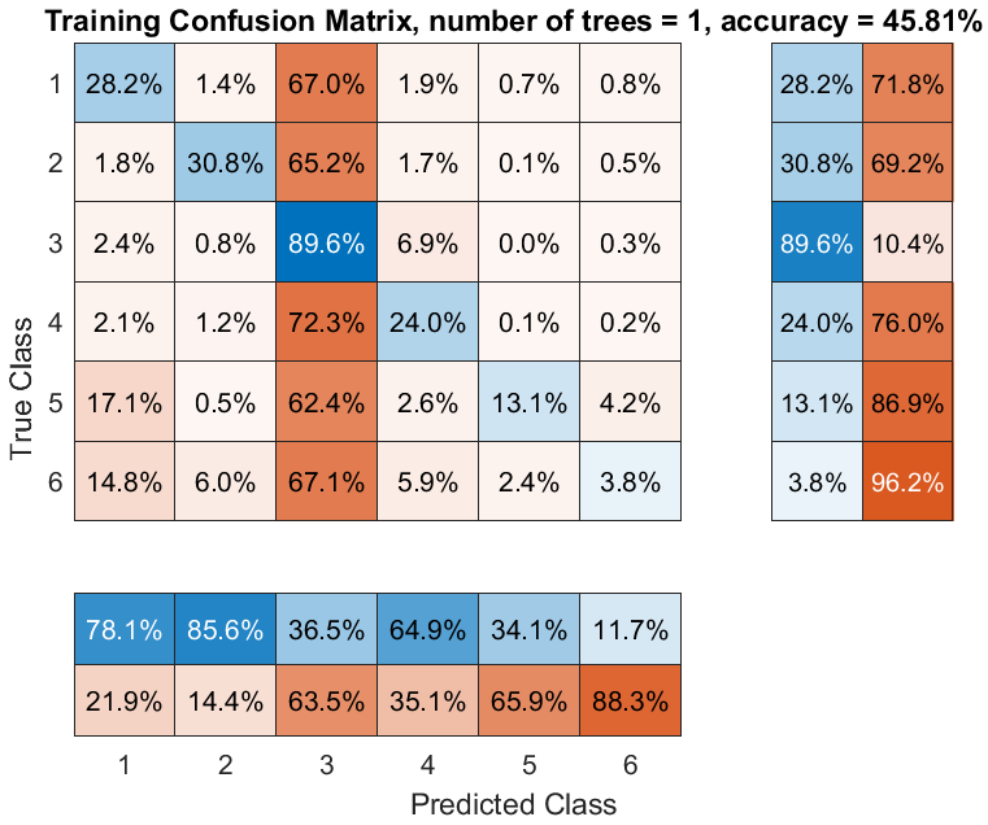

Figure 1.1: Confusion Matrix with 1 tree of training image using chessboard

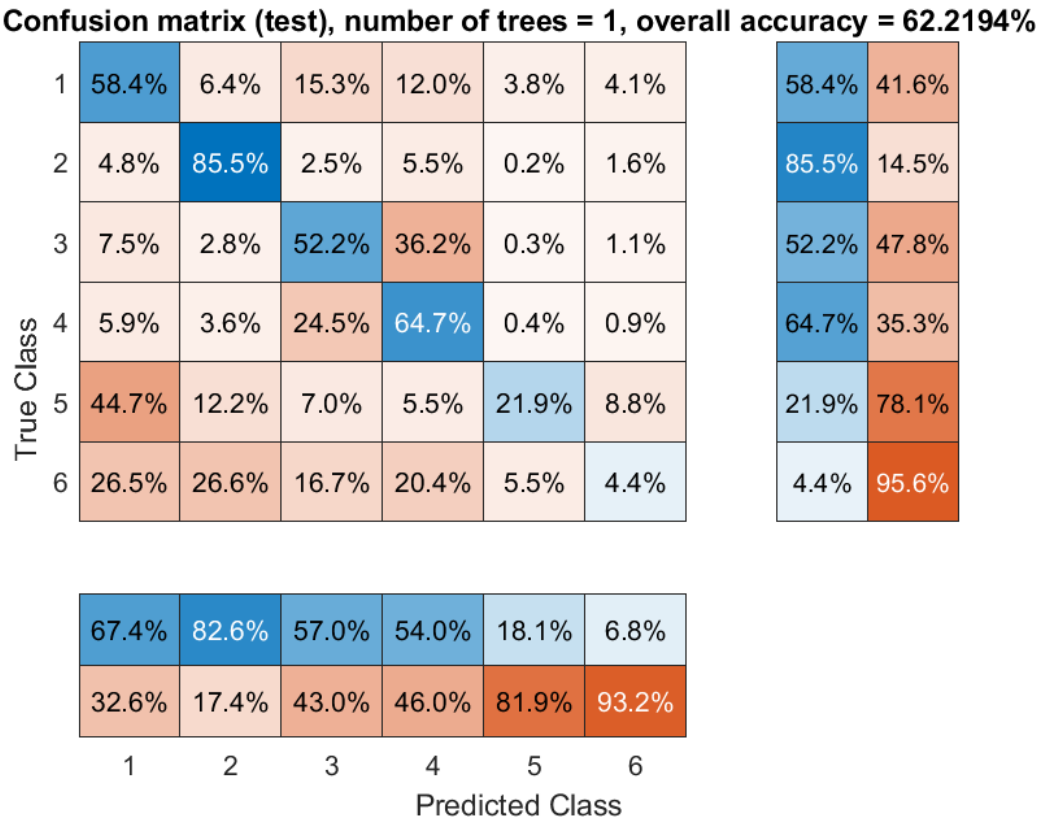

Figure 1.2: Confusion Matrix with 1 tree of test image using chessboard

1.2) SLIC

The resulting overall accuracy for the test image is comparable to the one achieved using chessboard. In case of the the training image there seems to be a strong bias again but towards class 1 this time. As before more trees are needed to produce better results.

1.3) Discussion

For both chessboard and SLIC the main reason for the training images low accuracy is a very strong bias towards a specific class. A bias that almost completely disappears for the test image. The easiest explanation for this behaviour would be overfitting to the Training Dataset. Specific features associated with class 3 or class 1 might be over represented in the training data resulting in a strong bias. Considering only one tree is used such an outcome would be realistic. The same features might be less common in the test image resulting in better accuracy. Using SLIC instead of chessboard segments the data differently resulting in the bias shifting towards a different class. The easiest way to try to avoid this would be to use more trees ensuring a more representative dataset. Using ten trees causes the bias to disappear as shown in the following section.

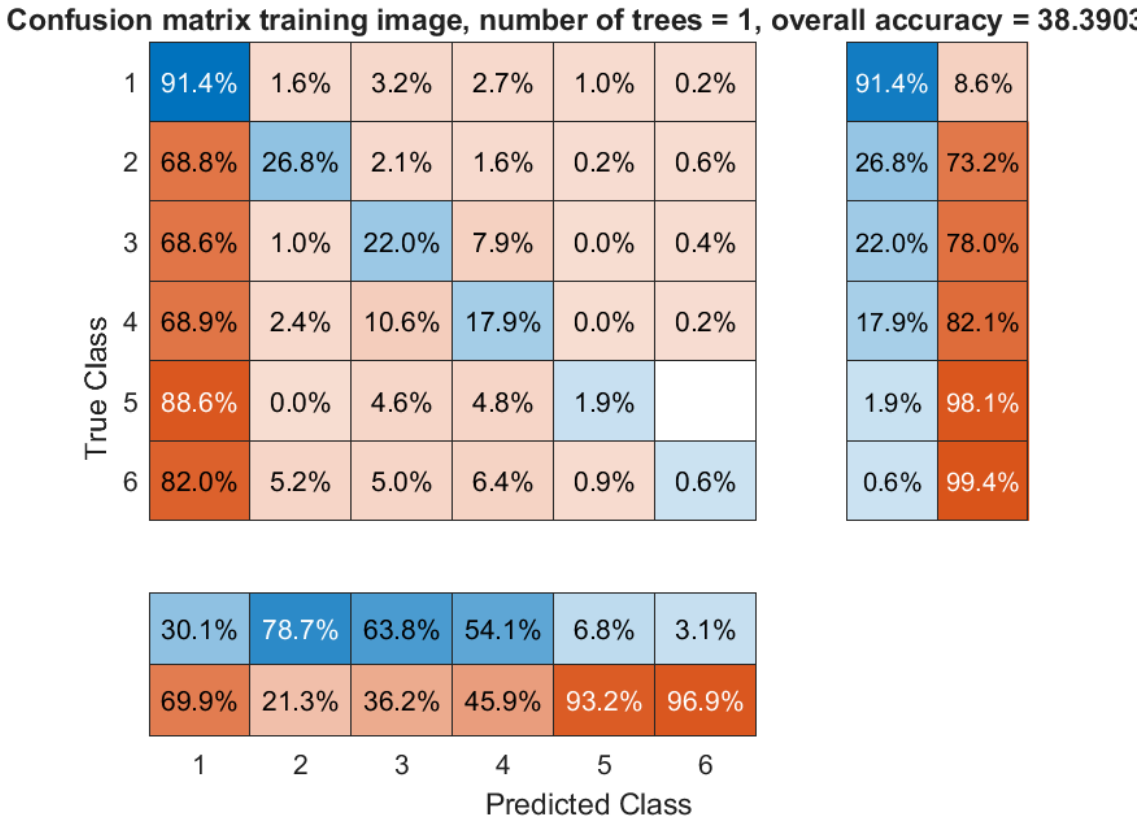

Figure 1.3: Confusion Matrix with 1 tree of training image using chessboard

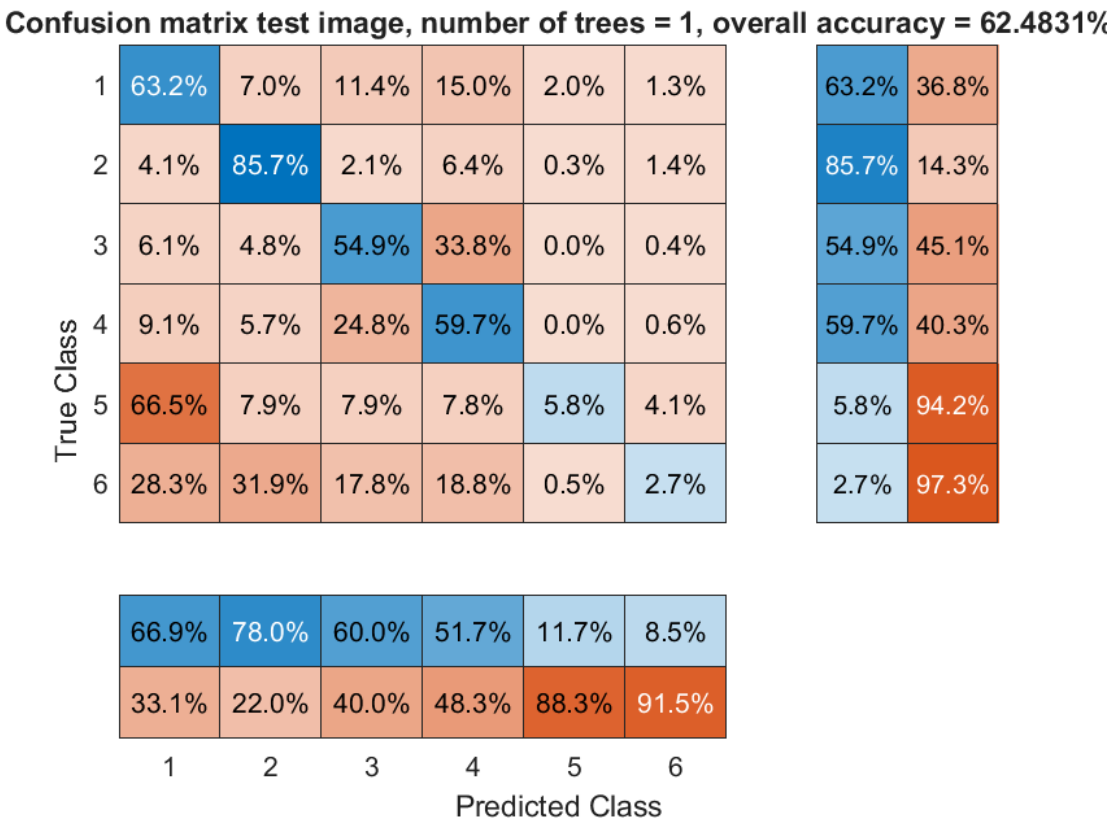

Figure 1.4: Confusion Matrix with 1 tree of test image using chessboard

2) Using ten trees.

2.1) Chessboard

Increasing the number of trees to ten significantly improves overall precision. The accuracy of the training data increases to 76 percent surpassing the test data accuracy of 68 percent. The distributions for the test and training images get a lot more similar with comparable accuracy for all classes. At this stage the robustness is quite good. Cars being the most diverse and hardest to recognise class has the least accuracy while the most homogeneous class buildings has the highest.

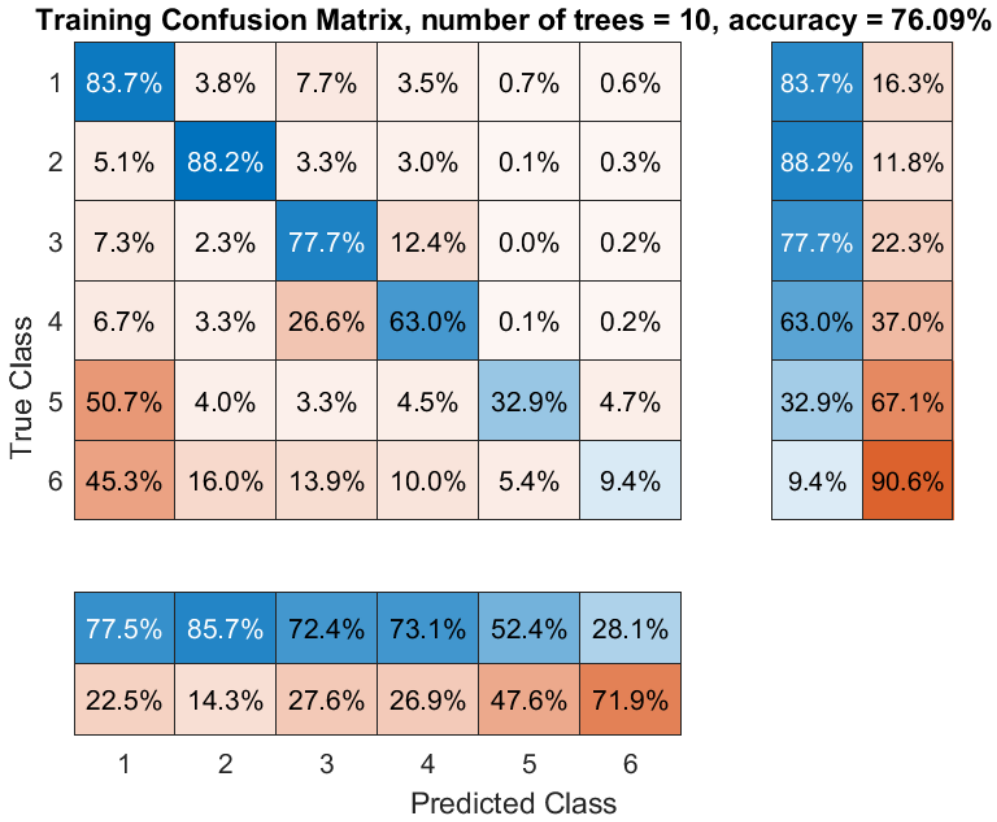

Figure 1.5: Confusion matrix training data for 10 trees using chessboard

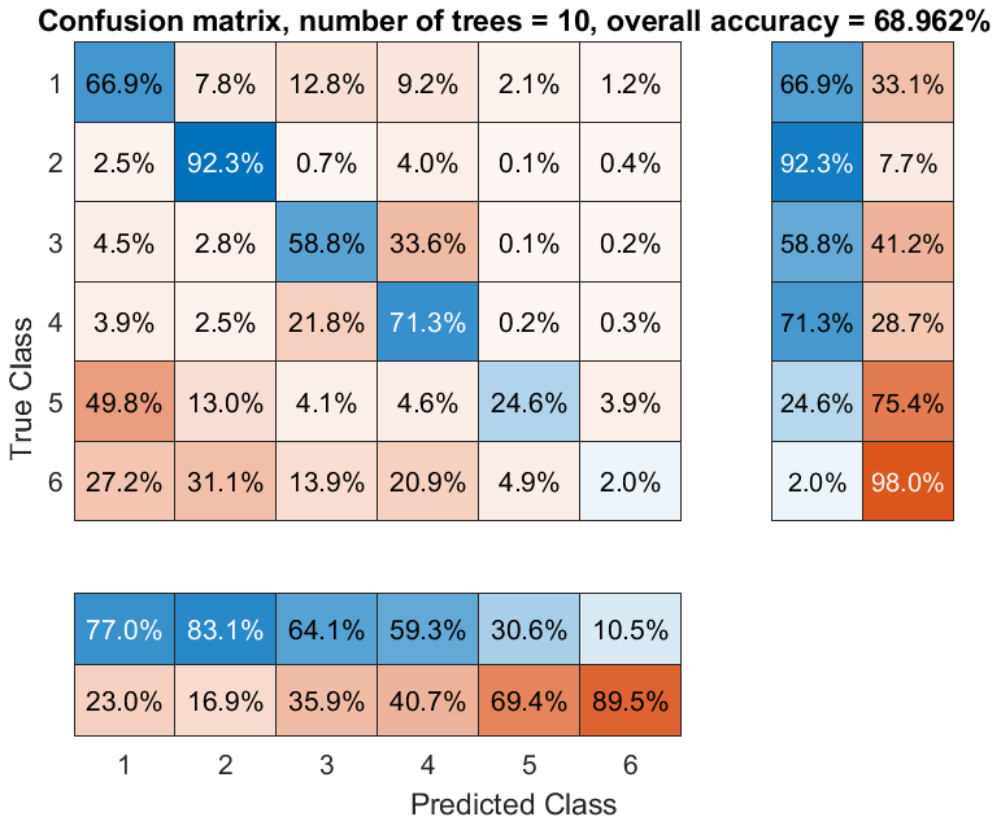

Figure 1.6: Confusion matrix test data for 10 trees using chessboard

2.2) SLIC

The resulting overall accuracy is similar to the one achieved using chessboard. The only noticeable difference is the 68 percent accuracy achieved on the training image clearly lower than the 76 percent accuracy achieved using chessboard. This is probably a random affect caused by the still relatively low number of trees. its noticeable that cars get recognised even less frequently when using SLIC compared to chessboard. Using chessboard at least 24.6 percent of cars got recognised correctly. Using SLIC this value drops to 1.8 percent.

2.3) Discussion

Chessboard performs significantly better at recognising cars compared to SLIc. This might be caused by the relatively small chessboard segments (15 by 15 pixels) allowing cars to be resolved better. The larger SLIC superpixel might be dominated by other labels even if a car is in the segment. Additionally cars are a inhomogeneous class with many different colours. in most cars being classified as impervious surfaces nearly 80 percent end up in the building class.

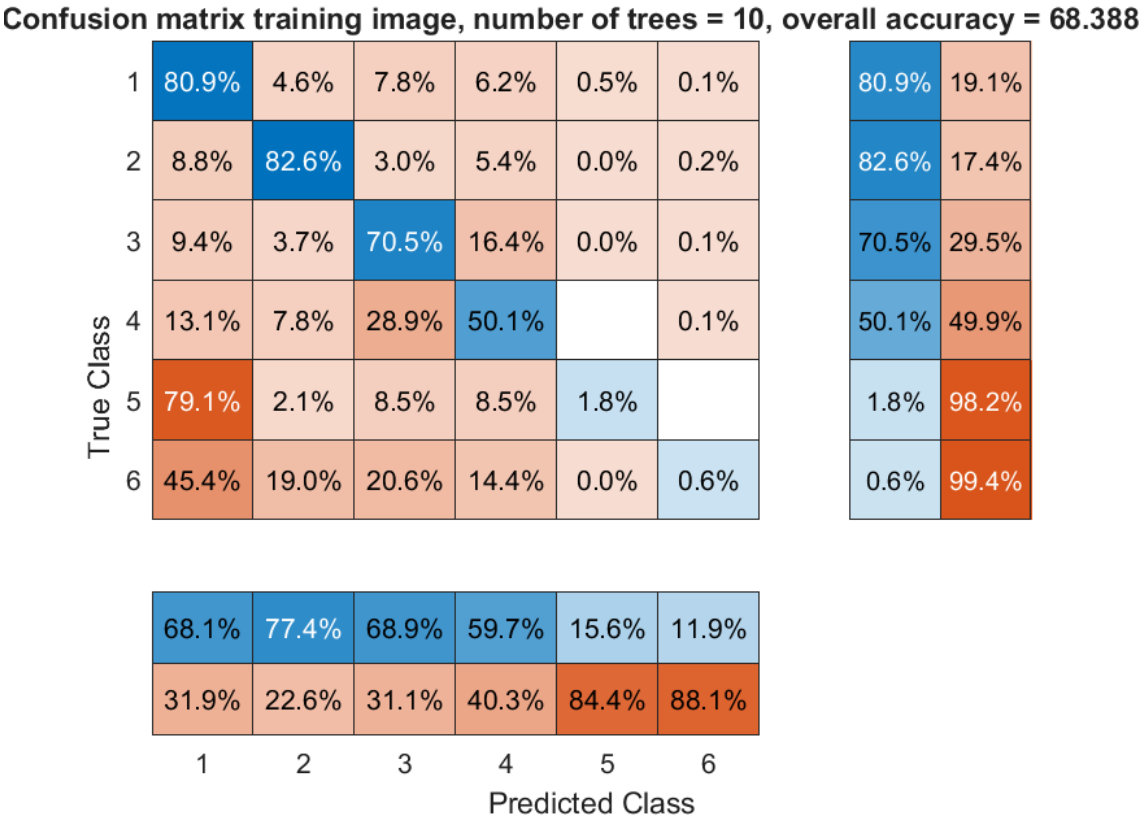

Figure 1.7: Confusion matrix training data for 10 trees using SLIC

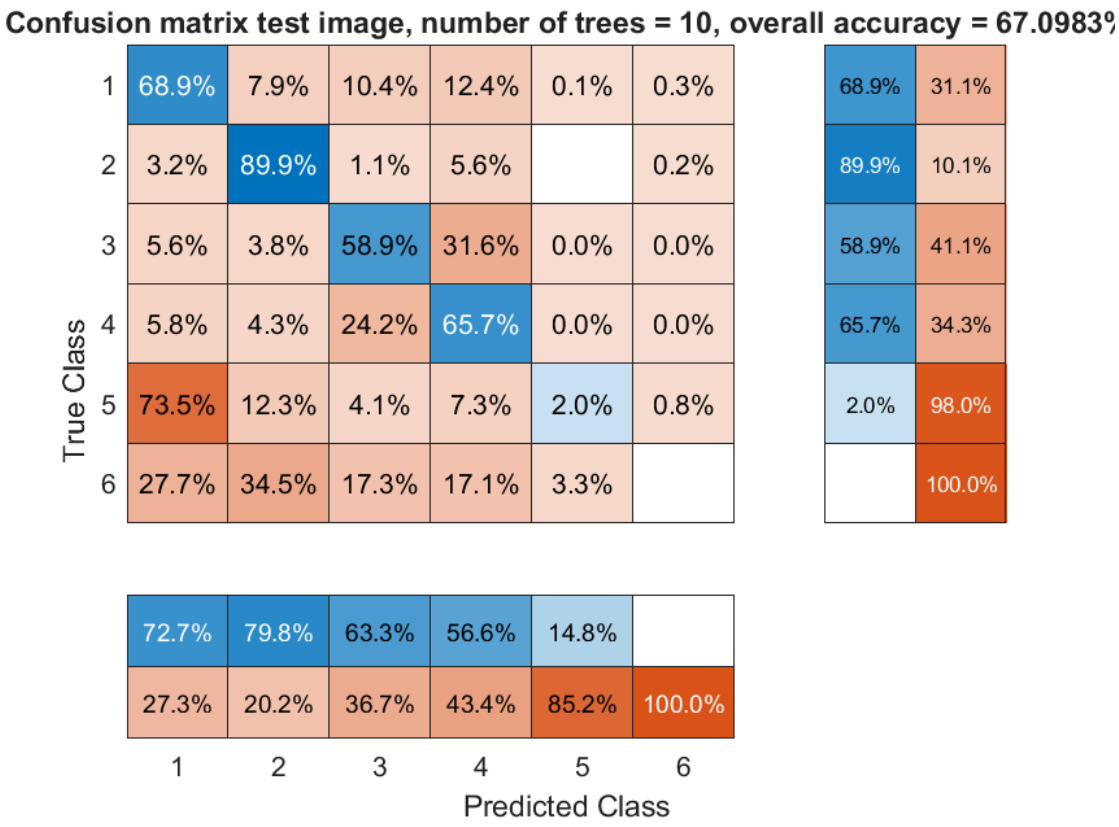

Figure 1.8: Confusion matrix test data for 10 trees using SCLIC

3) Resulting images

3.1) Chessboard

3.1.1) One tree

As the low overall accuracy of the confusion matrix already implied using one tree for the training data results in a barely recognisable image. as almost everything is classified as class 3 low vegetation. The image for the test data is better. buildings roads and areas of vegetation are clearly discernible. As expected using chessboard there is a lot a lot of random noise.

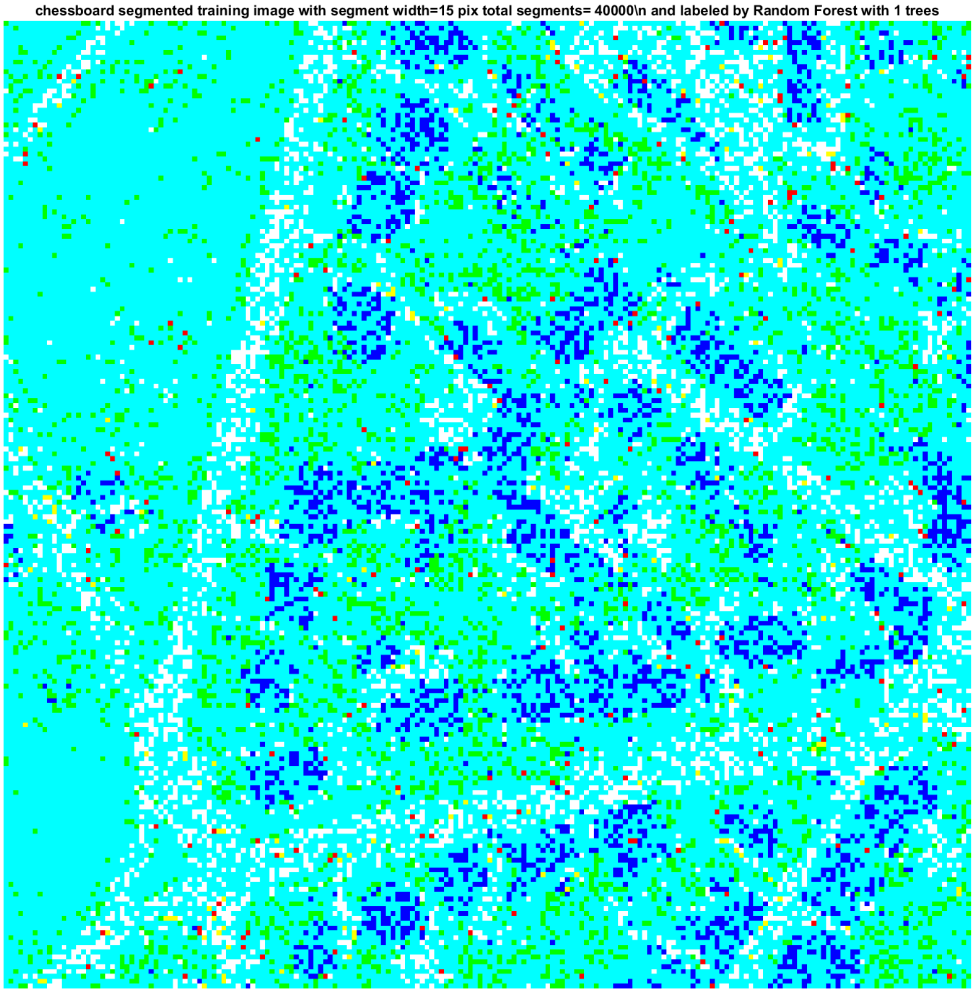

Figure 1.9: one tree training image using chessboard

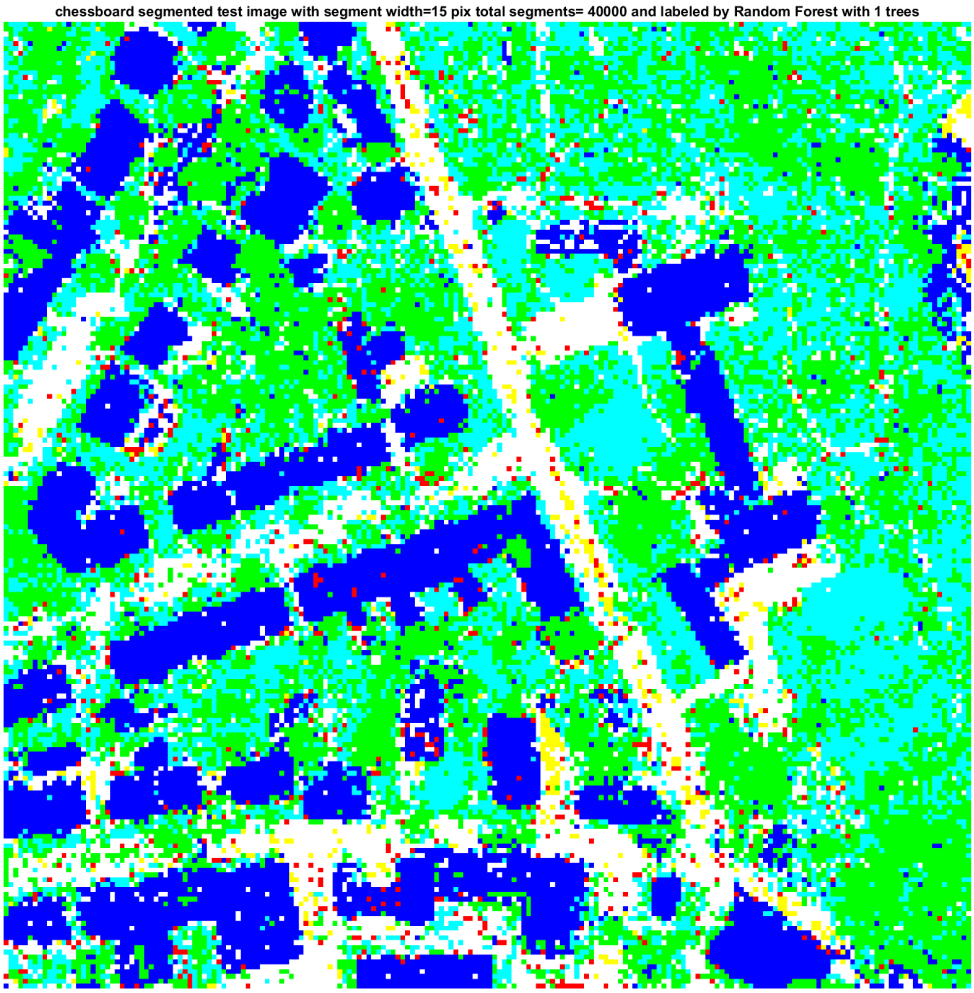

Figure 1.10: one tree test image using chessboard

3.1.2) Ten trees

Using ten trees the quality of the images is very similar Roads buildings and vegetation are clearly visible. Cars as expected are still a big problem as they get mistaken for clutter most of the time. There is less noise than before but its still clearly present. especially when compared to the images produced suing SLIC.

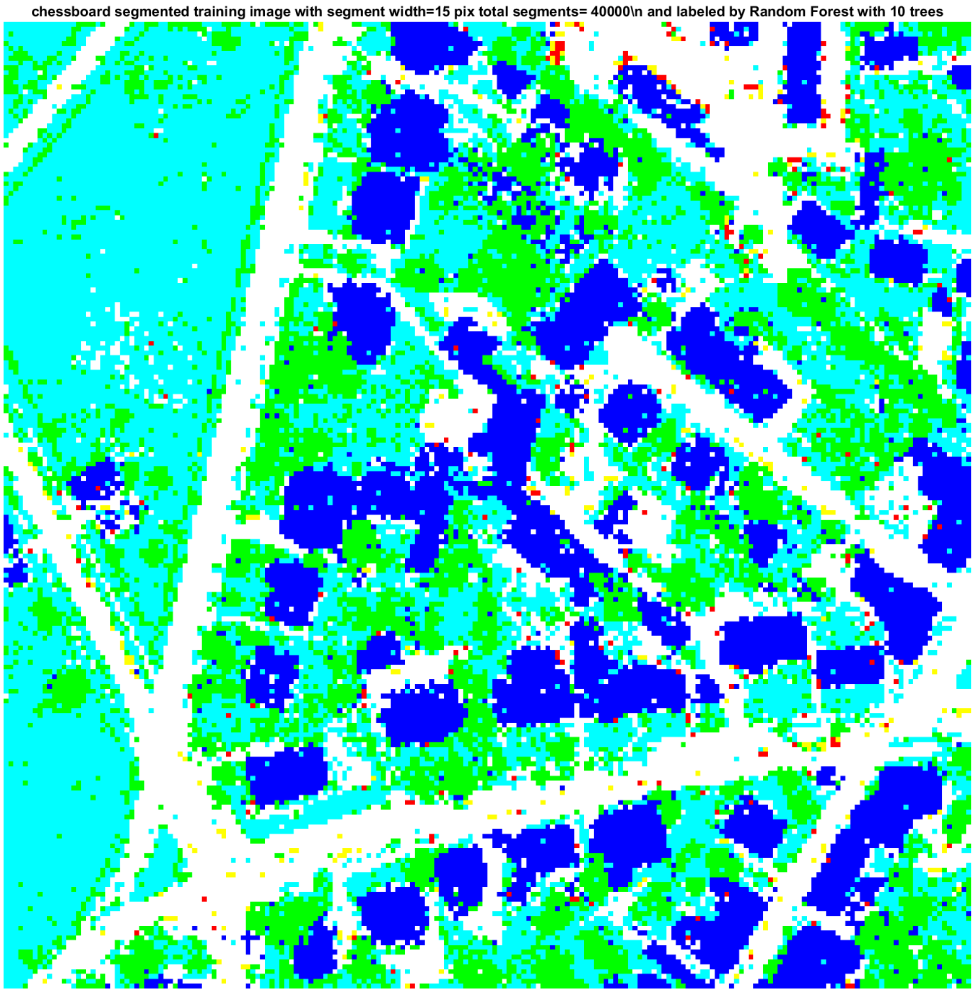

Figure 1.11: training image for 10 trees using chessboard

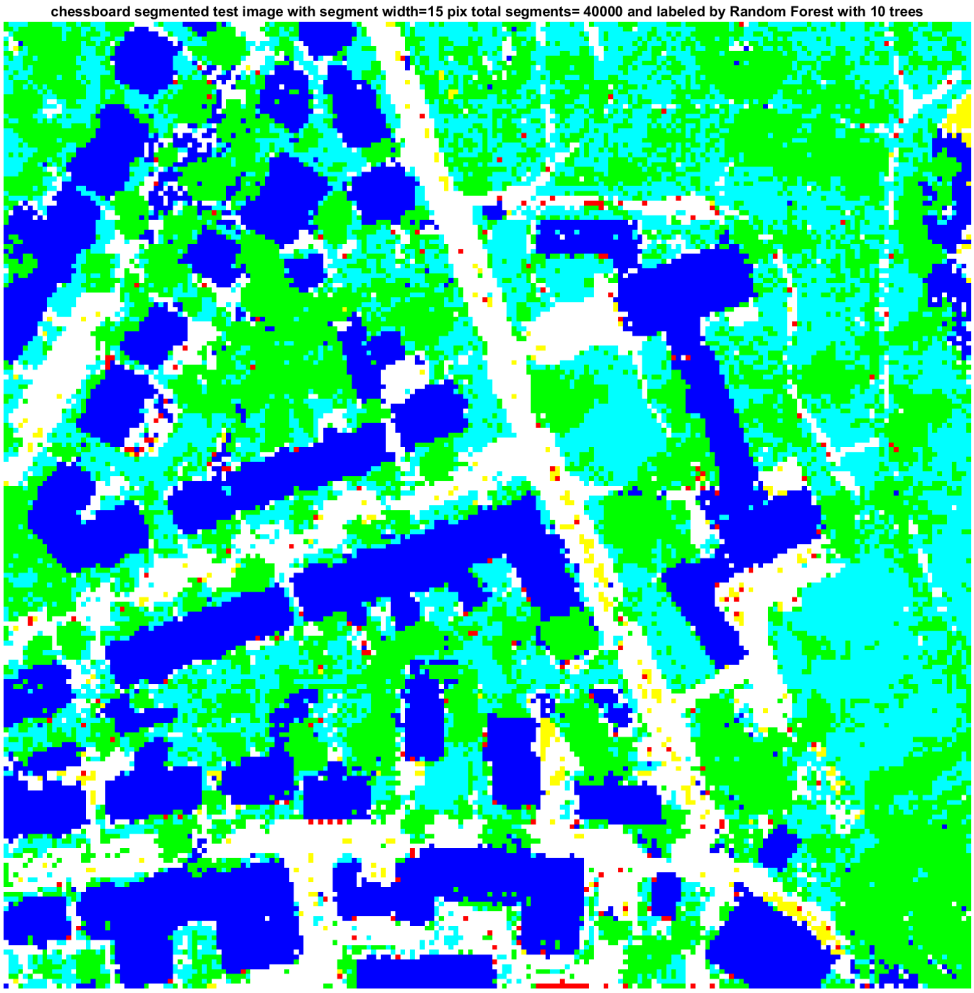

Figure 1.12: test image for 10 trees using chessboard

3.2) SLIC

3.2.1) One tree

Using only one tree the image produced using the training data is barely recognizable. The main difference to the chessboard algorithm is that the strong bias to class 1 impervious surfaces results in a mostly white image. Because SLIC was used instead of chessboard, the image is more homogeneous. Geometric shapes are visible and there is less clutter. The image resulting form the test data is comparable in quality to the one produced using chessboard. The main difference is that once again the image produced using SLIC is more visually coherent.

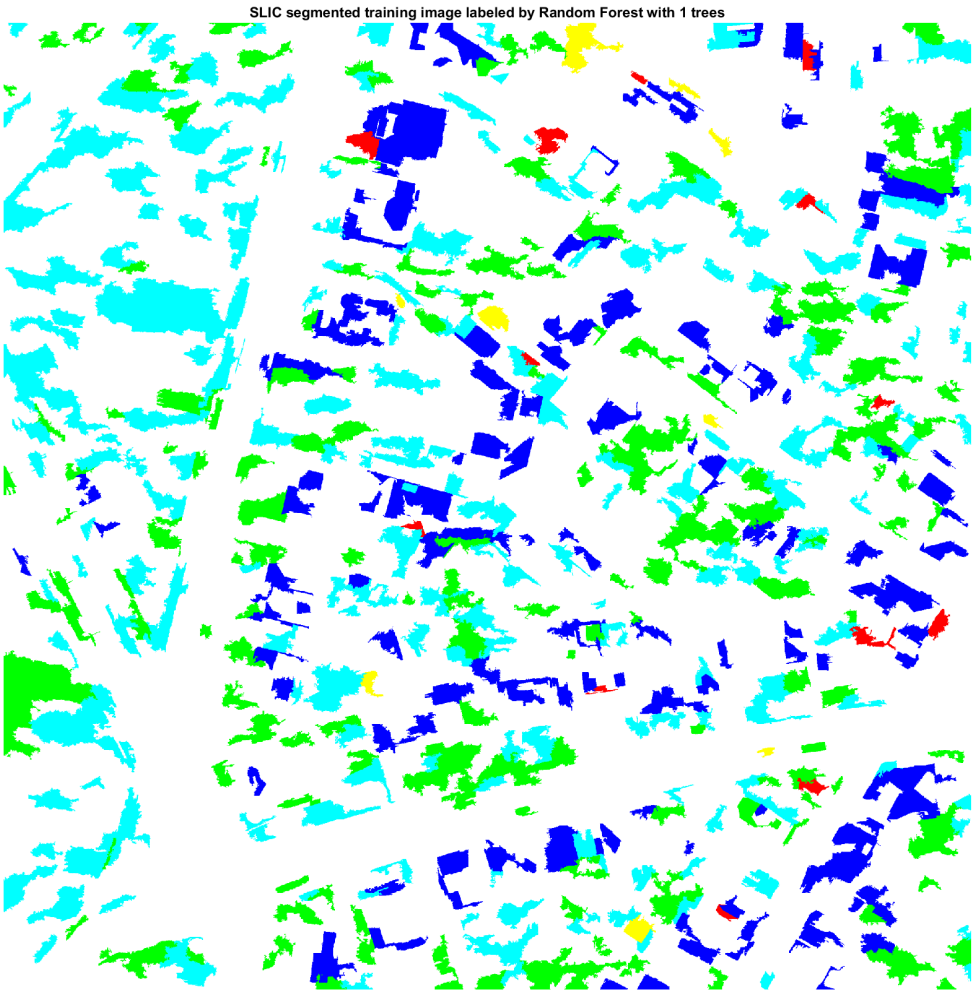

Figure 1.13: one tree training using SLIC

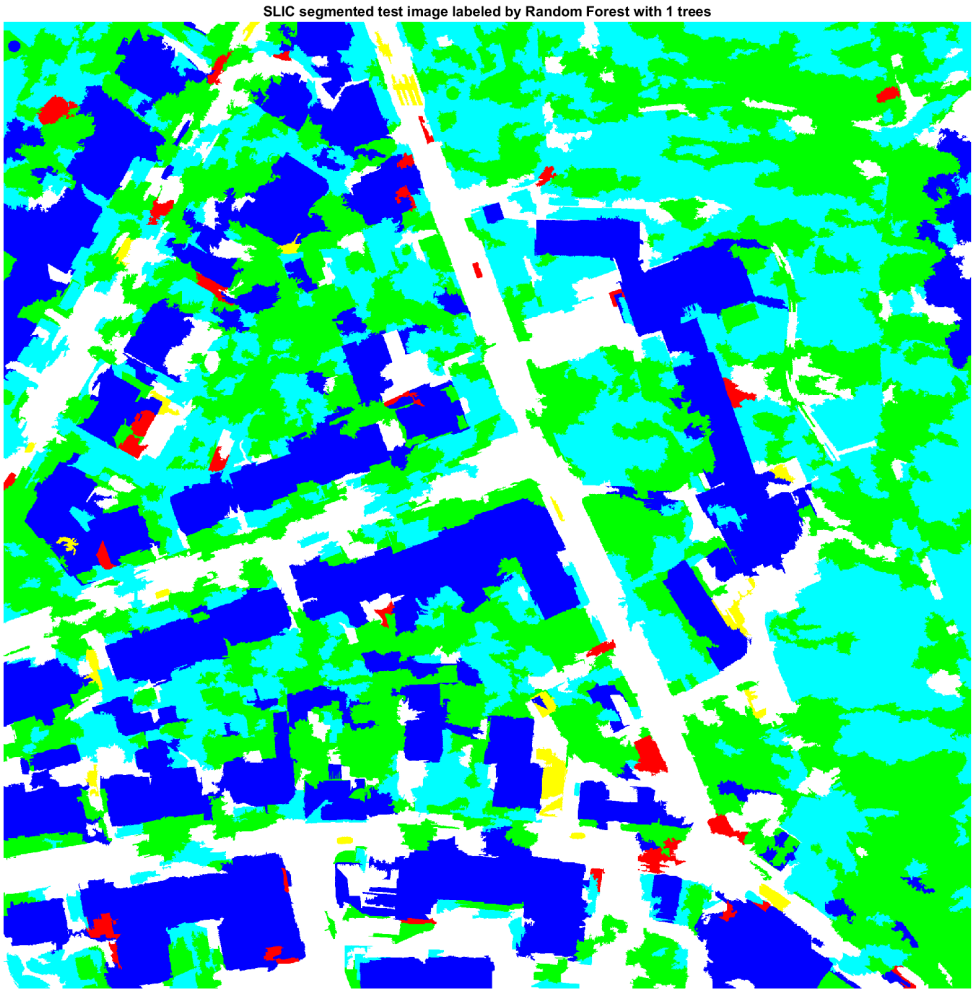

Figure 1.14: one tree test using SLIC

3.2.2) Ten trees

The SLIC based images have less noise but as pointed out in the confusion matrix section there are no cars visible on the roads, as they don't get recognised In terms of quality the images produced from the training and test data are very similar.

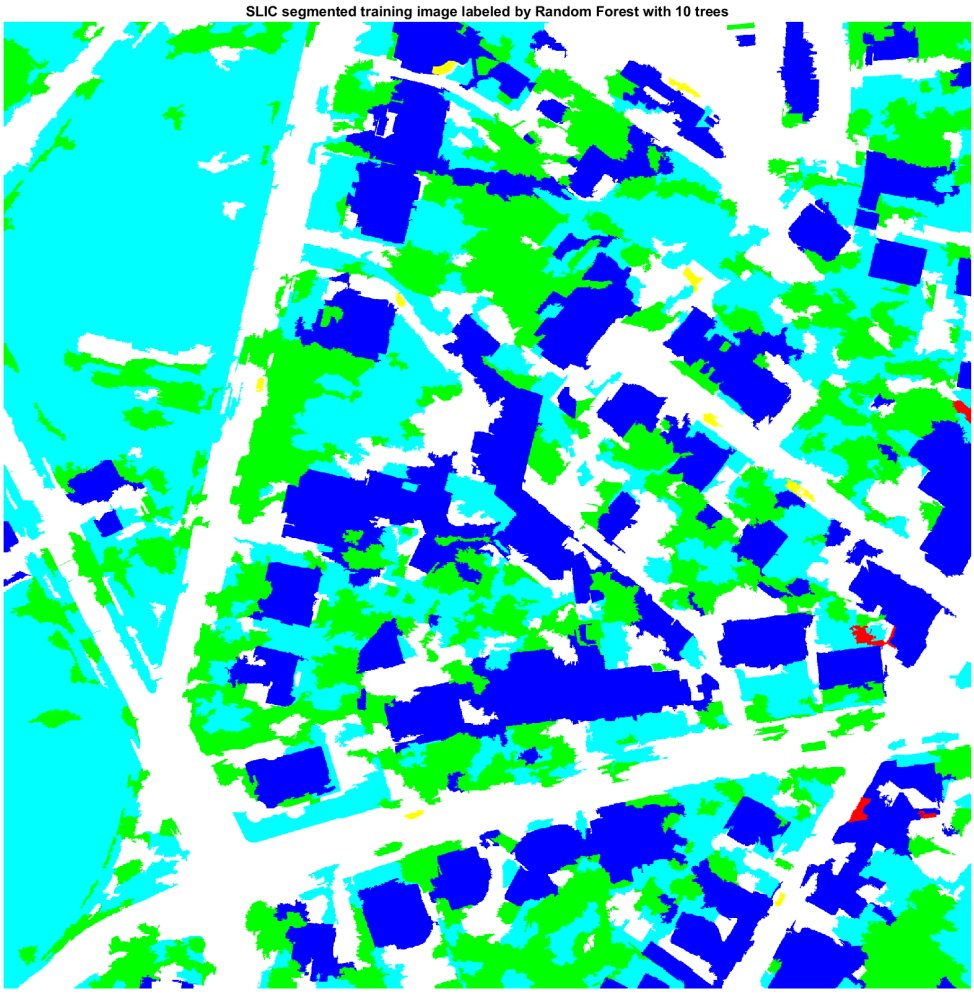

Figure 1.15: training image for 10 trees using SLIC

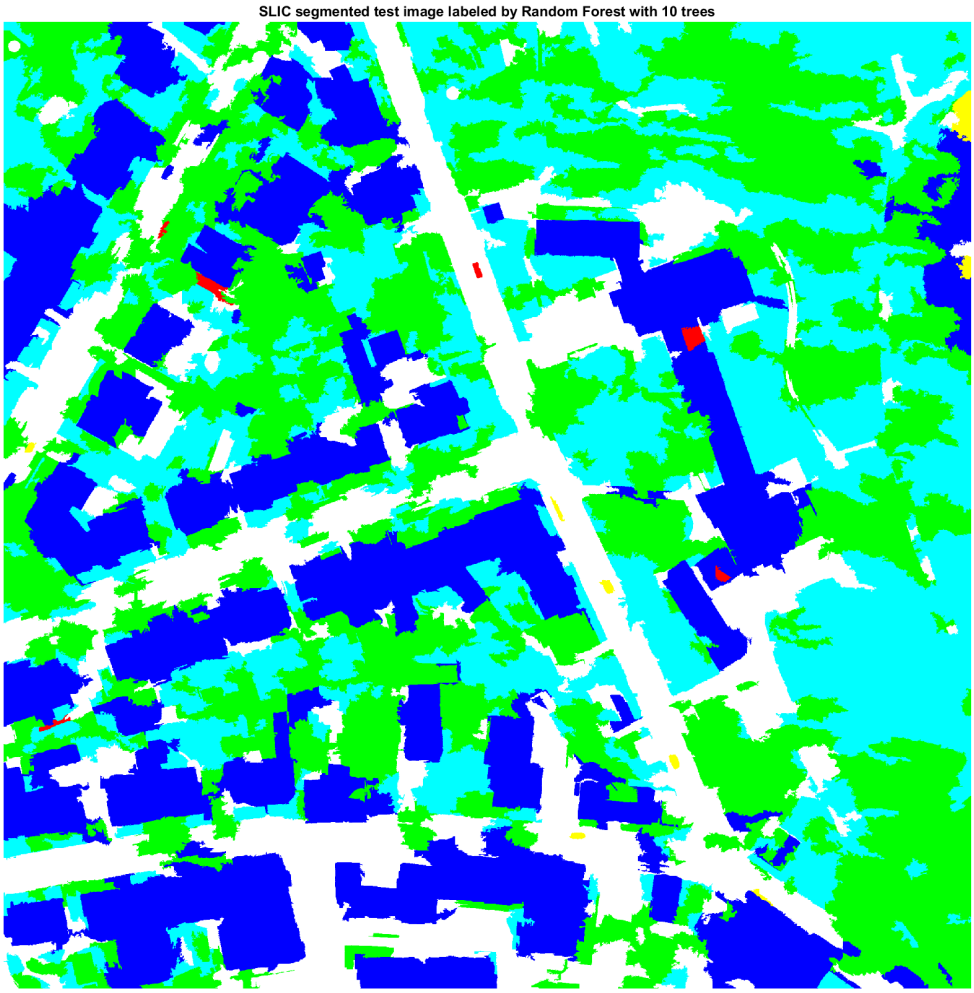

Figure 1.16: test image for 10 trees using SLIC

4) Finding the optimal amount of trees

4.1) Iterative

The optimal amount of trees in our case is 5 for chessboard and 10 for SLIC, as these value show the best test accuracy. After these, only the train accuracies increase, thus the models overfit. Moreover, the computational cost increases unnecessarily.

It is interesting to note that the points of optimal test accuracy are also the ones with the lowest computational cost (kinks). Clear reasons for this have not yet been identified.

4.2) Out-of-Bag-Error

While the iterative method shown above produces precise results training all the trees requires a lot of computation power. Using the out-of-bag-estimate provides a good first assessment at a considerably lower computational cost. This works by using the training data not chosen for the bootstrap sample. This process is repeated for each tree creating the OOB set. This is then used to to test the model.

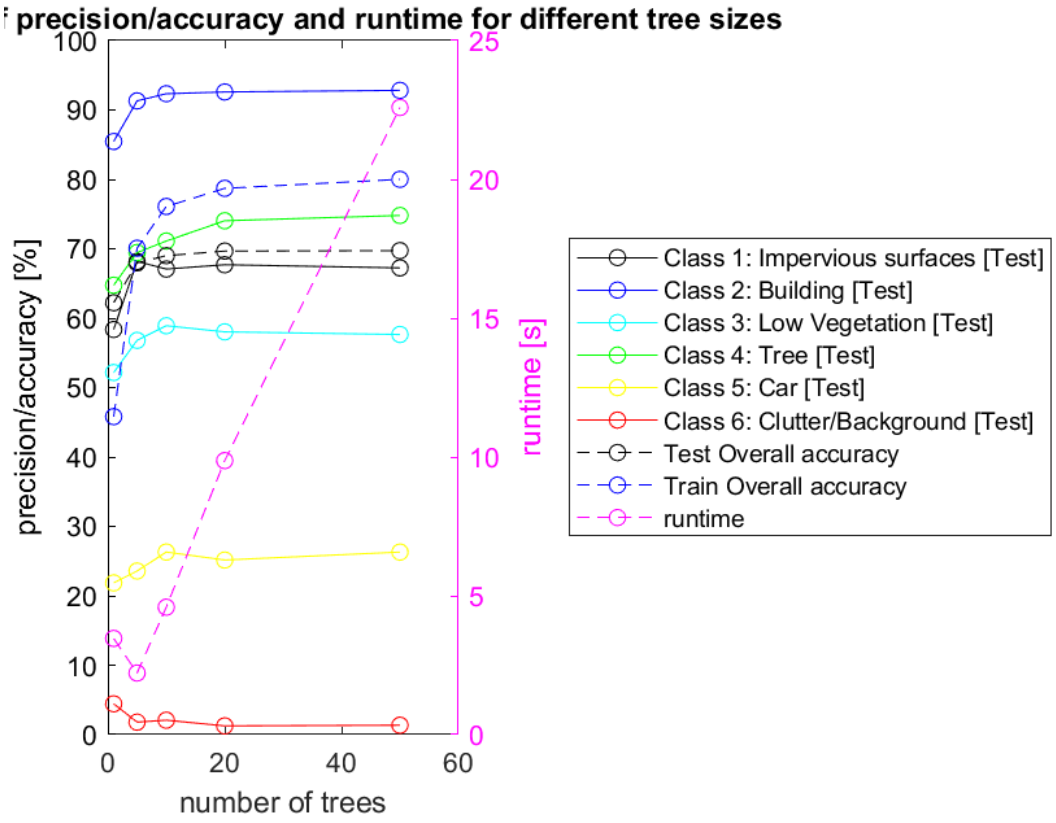

Figure 1.17: accuracy and (test) runtime depending on tree size chessboard

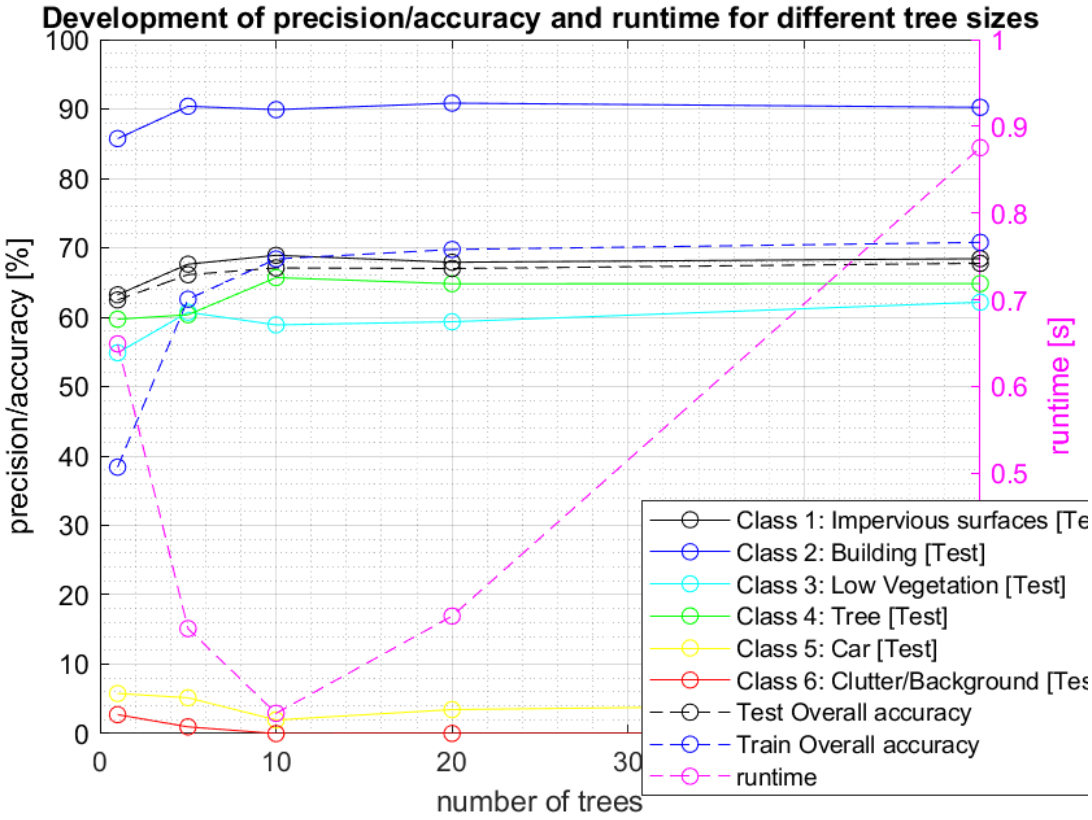

Figure 1.18: accuracy and runtime (test) depending on tree size SLIC

Questions

1) Explain and discuss the differences between Bagging and Boosting.

Bagging (Bootstrap Aggregating) and Boosting are both ensemble learning techniques but with distinct methodologies and objectives.

Bagging: This method involves generating multiple versions of a training dataset using bootstrapping (random sampling with replacement) and then training separate models on each dataset. The predictions of these models are combined through averaging (for regression) or voting (for classification) to produce a final output. The primary aim of bagging is to reduce variance and prevent overfitting by averaging out the noise in the predictions.

Boosting: In contrast, boosting sequentially trains models, where each new model focuses on the mistakes of the previous ones. This is done by assigning higher weights to misclassified instances, making the model pay more attention to difficult cases. The predictions are then combined to form a weighted sum, emphasizing the strengths of each model. Boosting aims to reduce bias and variance by iteratively improving the model's performance on the training set.

2) What is the main idea behind Random Forest?

The main idea behind a Random Forest is to create a 'forest' of decision trees, each trained on a different subset of the data and features, to improve the overall predictive accuracy and robustness. Random Forests leverage the power of multiple decision trees to reduce the risk of overfitting and increase generalizability. Each tree is built using a random sample of the data, and the best feature for splitting at each node is selected from a random subset of features. The final prediction is made by aggregating the outputs of all individual trees, typically through majority voting for classification or averaging for regression.

3) What is controlled by the depth of the tree? Can you make it arbitrarily deep?

The depth of a decision tree controls the maximum number of splits from the root node to the farthest leaf. It determines how complex the model is and how well it can capture the intricacies of the data. While increasing the depth can lead to a model that fits the training data more closely, making a tree arbitrarily deep can result in overfitting, where the model captures noise rather than the underlying pattern. Therefore, while theoretically possible, making a tree arbitrarily deep is not practical as it compromises the model's ability to generalize to unseen data.

4) What is controlled by the number of trees within a Random Forest?

The number of trees in a Random Forest controls the ensemble's robustness and the stability of the predictions. More trees typically lead to better performance as they reduce the model's variance and improve its ability to generalize to new data. However, after a certain point, adding more trees yields diminishing returns in terms of accuracy and can lead to increased computational cost. It is crucial to balance the number of trees to ensure computational efficiency without sacrificing the predictive power of the model.

5) How could you improve the achieved classification accuracy?

To improve classification accuracy, you could:

- **Increase the number of trees:** More trees can capture more patterns in the data.
- **Optimize hyperparameters:** Tune parameters such as tree depth, minimum samples per leaf, and feature subset size.
- **Use feature engineering:** Create or select more relevant features to improve model performance.
- **Reduce overfitting:** Implement techniques like cross-validation and pruning to prevent overfitting.

- **Use more data:** Expanding the training dataset can help the model generalize better.

6) What represents the entropy?

Entropy in the context of decision trees represents the measure of uncertainty or impurity in a dataset. It quantifies the amount of disorder or randomness and is used to decide how to split data at each node in the tree. A higher entropy indicates more mixed or uncertain data, while lower entropy indicates more homogeneity. The goal of decision trees is to partition the data in such a way that each subset has the lowest possible entropy, indicating more uniform and distinct classes.

7) Imagine you are given a two-class classification problem. Which split results in an entropy of zero? Make a sketch and draw the histograms for the sake of clarity and an example calculation resulting in an entropy of zero.

A split results in an entropy of zero when all the samples in each resulting subset belong to a single class. For a two-class problem, this means one subset contains only samples from class A and the other subset contains only samples from class B.

Example: If we split the dataset such that subset 1 contains all class A samples and subset 2 contains all class B samples, the entropy for each subset is zero because there is no uncertainty within each subset.

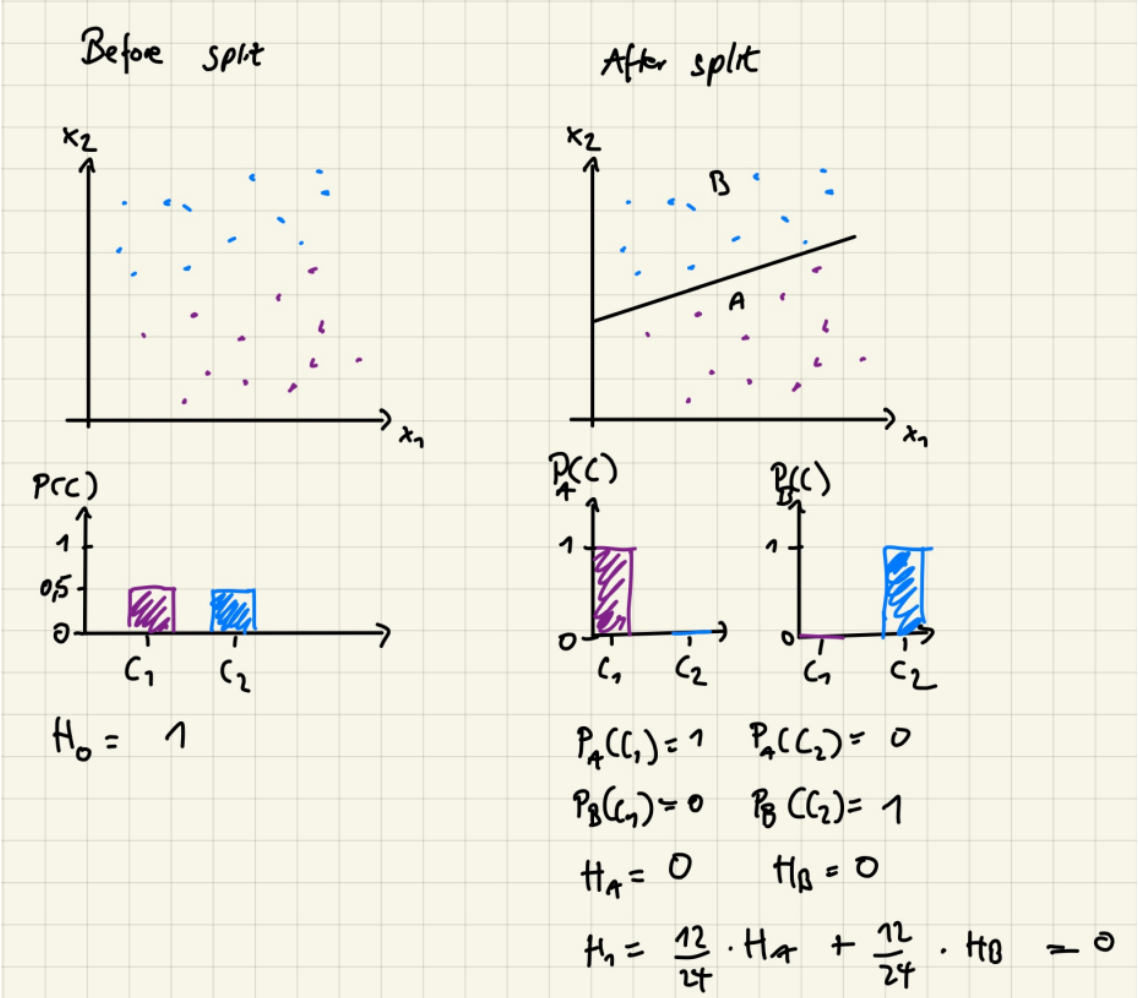

Figure 2.1: Sketch of histograms and entropy calculation for binary classification problem

8) Imagine you are given a two-class classification problem. What happens to the entropy when the probability of one class becomes:

a. Zero? When the probability of one class is zero, it means that all samples belong to the other class. This leads to an entropy of zero, indicating no uncertainty or impurity because the subset is homogeneous.

b. Equal to the other class probability? When the probability of one class is equal to the other class, the entropy is maximized. This represents the highest level of uncertainty because each class is equally likely, leading to maximum disorder and impurity.