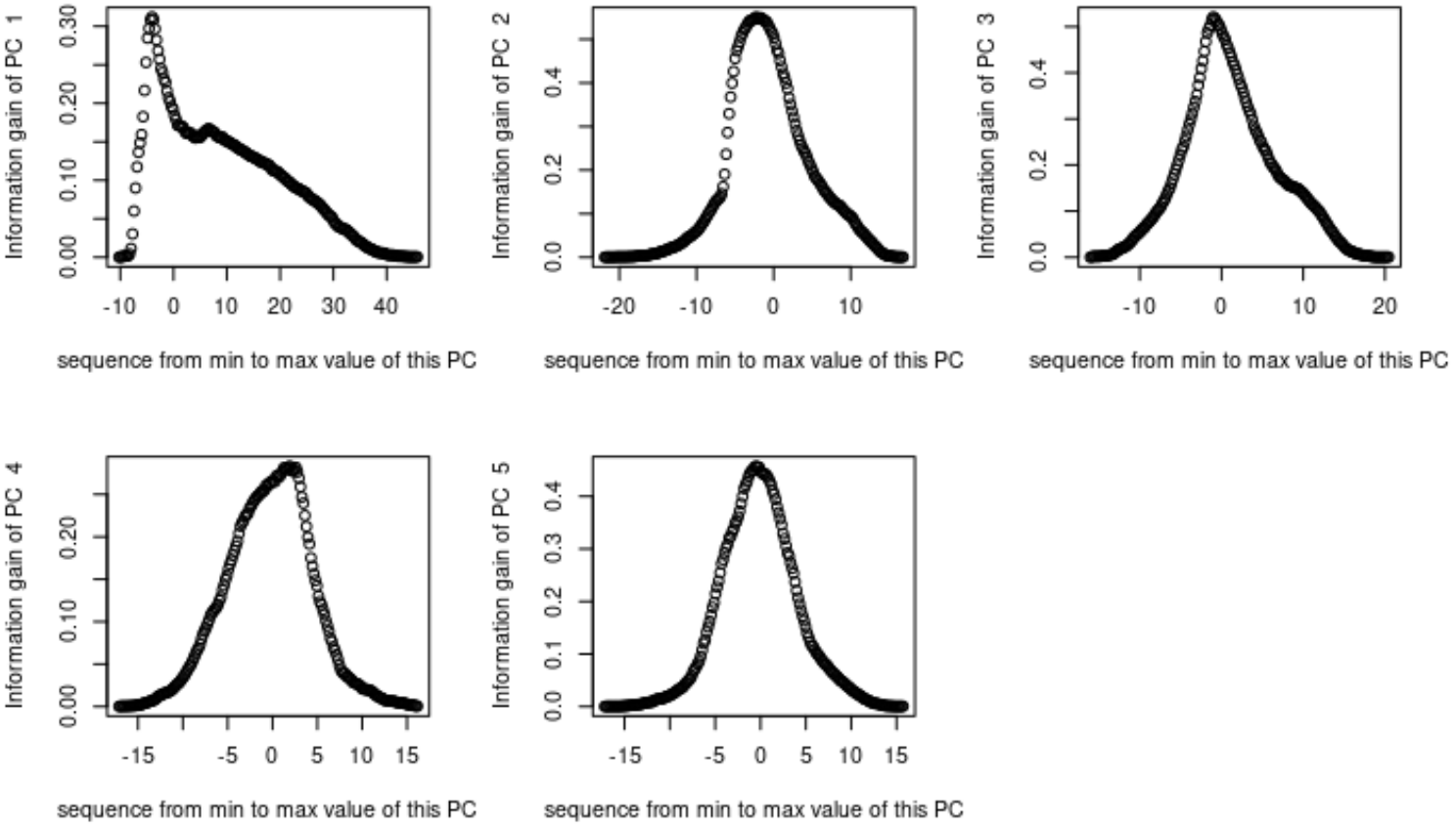
Group 4
Exercise 4 - Decision Trees and Random Forests
Ines Benomar, Vincenzo Coppola, Karol Szurkowski, Erik Kockar

4.1.1 - Compute the optimal decision point for the first 5 PCAs of a dataset (e.g. a single person) and compute the information gain associated to it (plot 5 graphs, one for each component, and show the highest information gain). See slides for how to compute information gain.

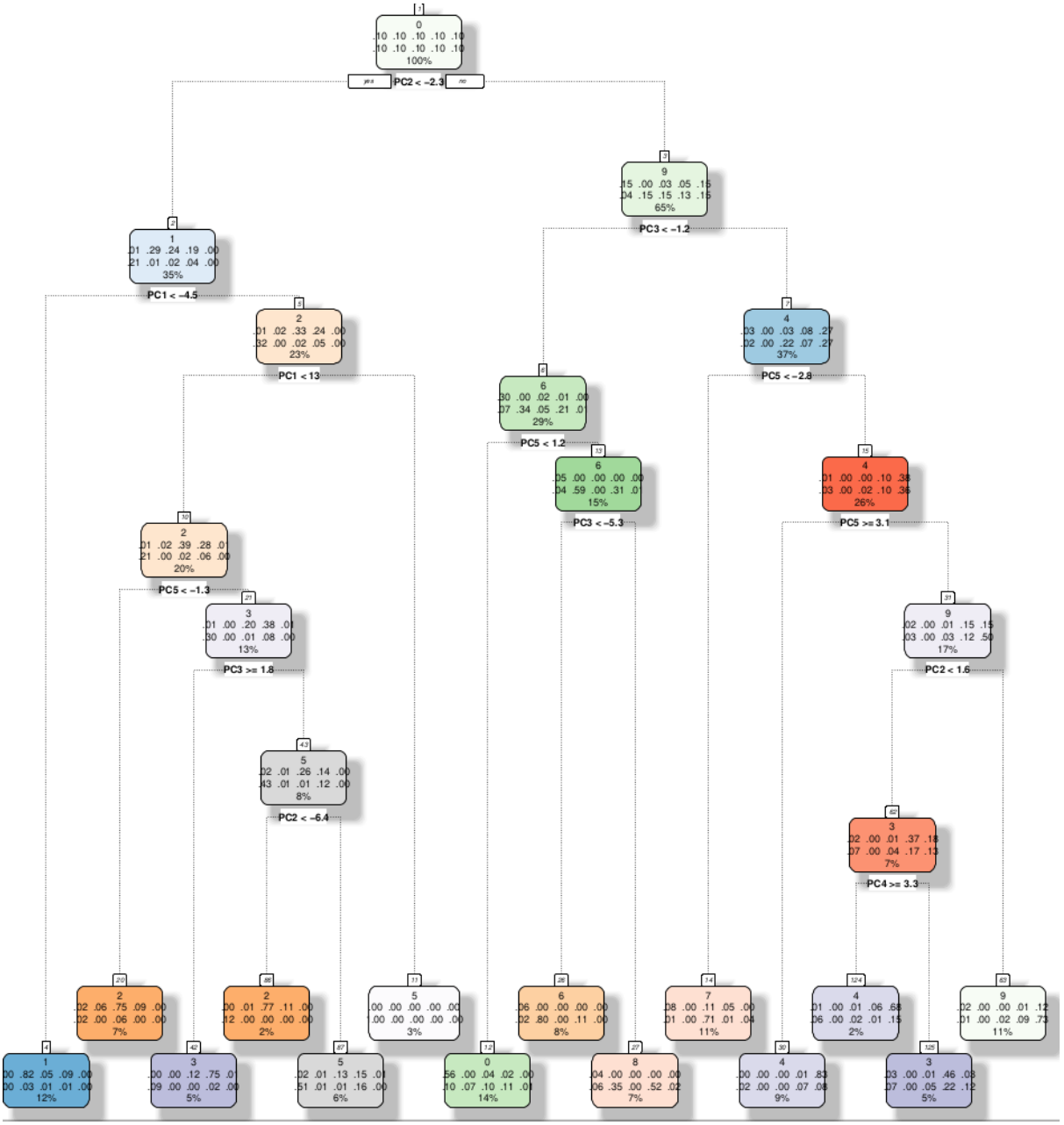


Understanding the information gain plots
x - sequence of min and max values of Principal Components (prcomp.x), we use it to find the best threshold to do classification
y - for every point of x.axis we are calculating the information gain, that we achieve with comparing the entropy before and after PCA decomposition.

We split data based on min and max values of PCA as thresholds-> split rows (means digits)

For every PCA, we compare the entropies to choose the optimal point which gives the highest information gain. So now for every PCA we have an optimal decision point (of splitting the main dataset) - that we have obtained with the PCA decomposition and entropy calculation.

4.1.2 - Compute a decision tree for the digit classification and visualize it.



To understand the plot we need to know few things:

top - is the inspected class
0.x is the probability of the Y% samples belonging to the each label (class)

Data is splitted by the thresholds calculated from the previous exercise. Here instead of "information gain" they are using the "Gini index".

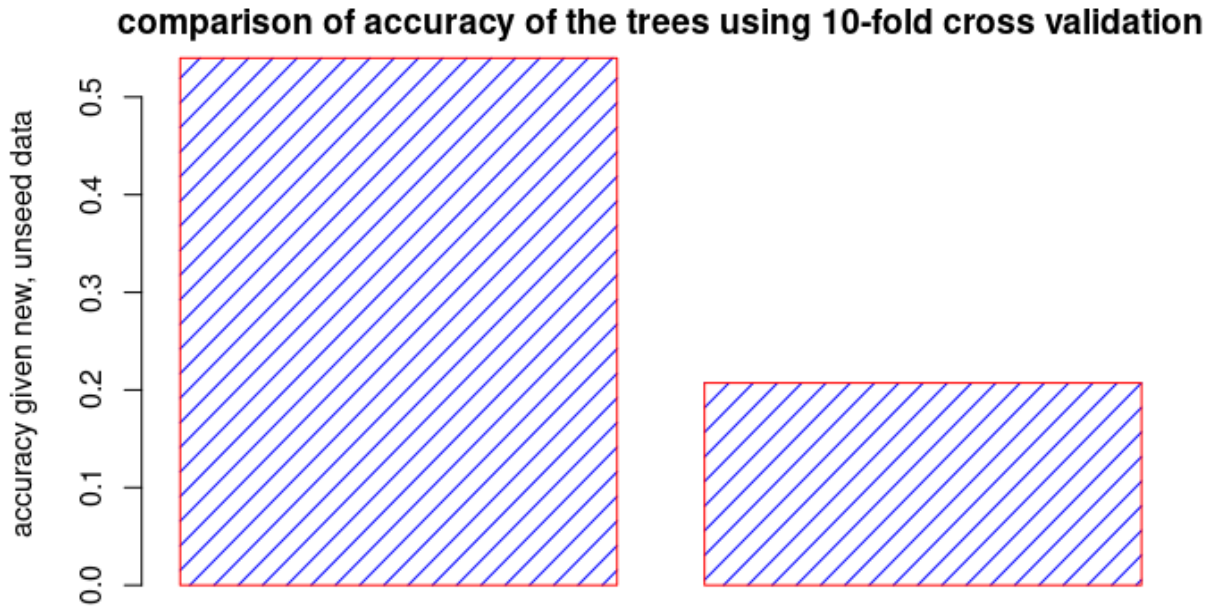
From the top, we start with 100% of the data -> 8000 of rows (digits) and we check what are the probabilities of the rows to be 0 (digit on the top of the rectangle— it shows the class with the highest probability for the data to belong to).

This 100% of samples has 0.1 probability to belong to every class, but the 0 is written on the top, because it is supposed to be the "highest" probability, then we want to split the data into two parts. We check which PCx has the highest "improvement" ~ related to the Gini index, which does the same thing as "Information Gain". So we basically split the data based on the threshold of the PCx with most importance.

Rows that are below the threshold go to the left side, and others go to the right -> in the next boxes you don't have 100% of data, but the percentage of the overall data that fits into the threshold filtering.

The whole tree shows us probabilities of classification of rows, based on the thresholding of the data with the entropy of the PC.

4.1.3 – Using the full data set (i.e. dataset from multiple people), evaluate a trained decision tree using cross validation. Try to train a tree with PCA, and without PCA (raw data). Discuss the important parameters.



Raw pixel data on the left and normalized PCs data on the right

4.2 Create a Random Forest classifier and evaluate it using cross validation. Discuss the critical parameters of "randomForest" (e.g., number and depth of trees)

Decision trees are built on the entire data set using all the predictor variables, whereas Random Forests are used to create multiple decision trees, such that each decision tree is built only on a part of the data set.

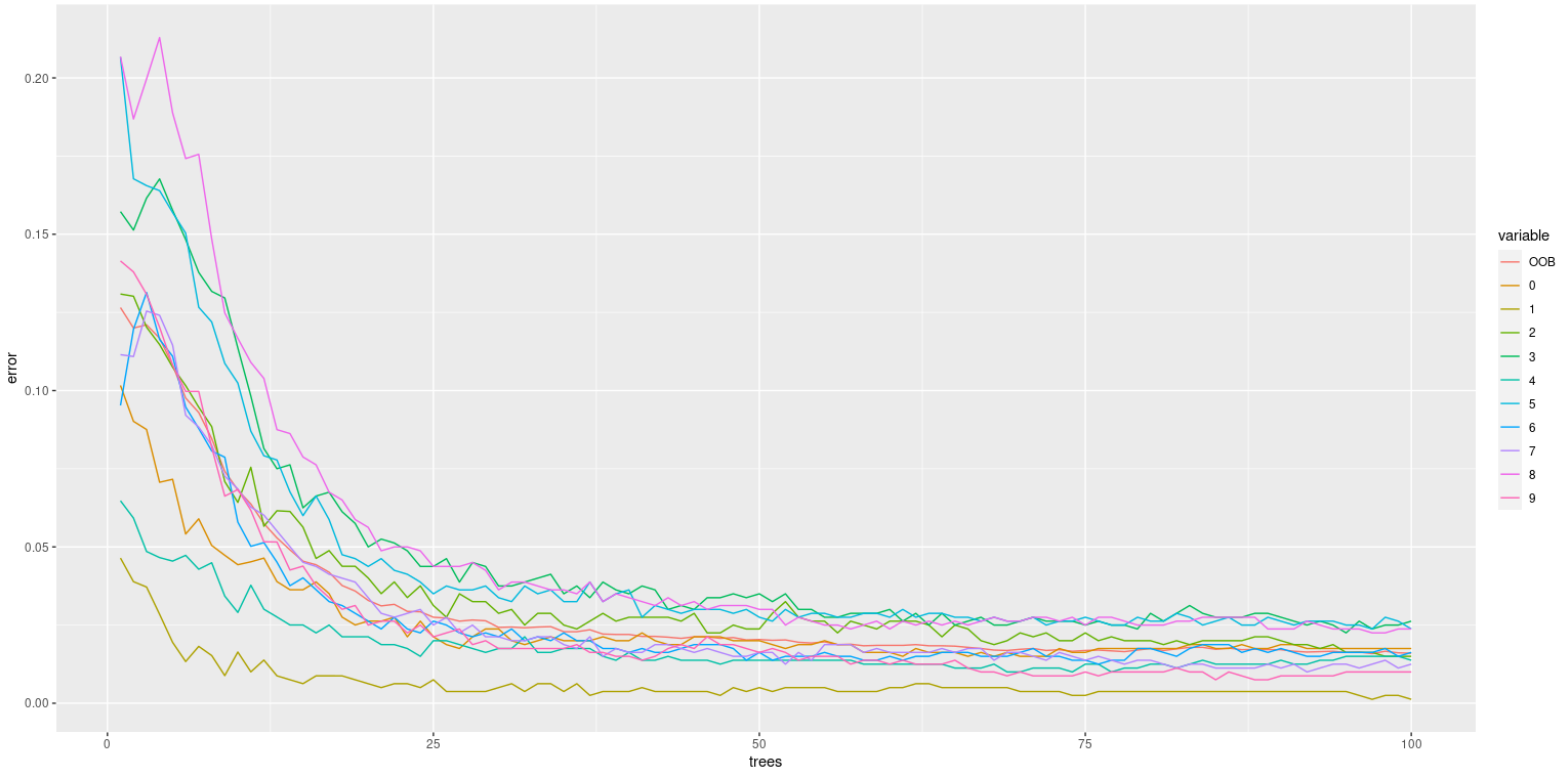
In the random forest approach, a large number of decision trees are created. Every observation is fed into every decision tree. The most common outcome for each observation is used as the final output. A new observation is fed into all the trees and taking a majority vote for each classification model.

Critical parameters to be considered when making random forests:

- **Number of trees:** Usually the higher the number of trees the better to learn the data. However, adding a lot of trees can slow down the training process considerably, and this should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
- **Depth of each tree in the forest:** The deeper the tree, the more splits it has and it captures more information about the data.

- **Minimum number of samples required to split an internal node:** This can vary between considering at least one sample at each node to considering all of the samples at each node. When we increase this parameter, each tree in the forest becomes *more constrained* as it has to consider more samples at each node.

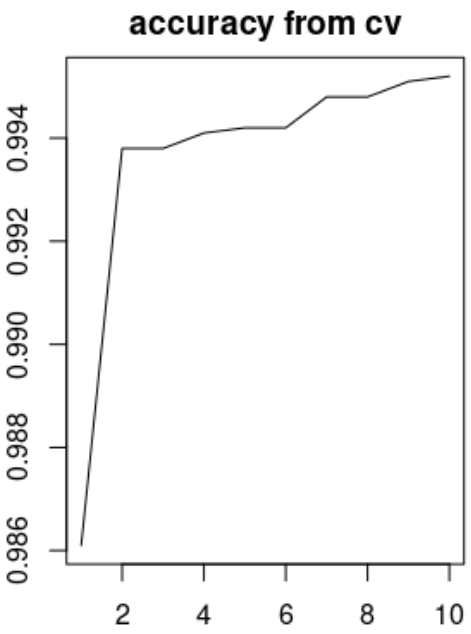
In the plot below, we can observe what is the misclassification for all of the cyphers as well as the OOB error.

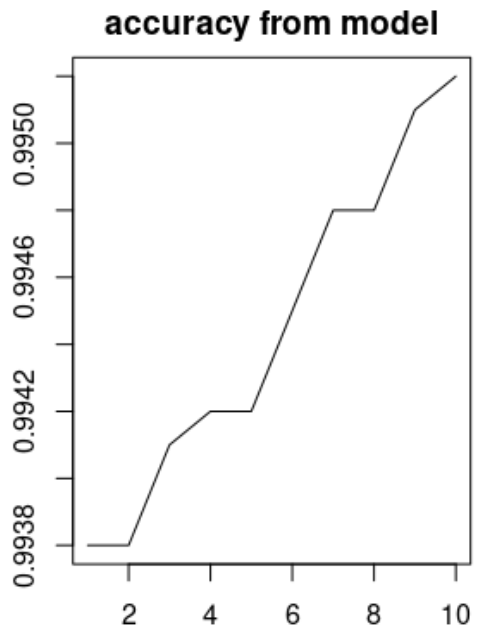


As seen in the plot, MSE converges after using 40 trees but even using as little as 20 trees give solutions worth consideration. Furthermore, we can observe that the smallest MSE was for the classification of the digit “1” and the largest was for the digit “8”.

OOB (Out of the bag) error can be defined as compared to the validation score is computed on data that was not necessarily used in the analysis of the model. What is worth mentioning is that the OOB error shows a sharp decrease sometime after ~5 trees.

To evaluate the performance of the model below we present results from a cross-validation test with 10 folds.





We know that by default, random forests do not use the full dataset for training (70% for training) and the rest for testing during classification. By using cross-validation we can ensure that all samples will appear in training and test sets, therefore eliminating any possible bias that may arise. As seen in the plots above, when using cross-validation the accuracy of the model, while still high in both cases, is moderately lower (starting at 0.986 with cv versus 0.9938 for the model).