Forest cover type classification using AdaBoost

Alessia Caputo
alessia.caputo@studenti.unimi.it

The experiment concern the development of an AdaBoost model for the classification of the Cover Type dataset. The first part of this report describes the AdaBoost model and the decision stumps. The subsequent section describes the dataset used and its processing. The last sections illustrates the results.

AdaBoost

Ensemble methods are used to form combinations of predictors that achieve a bias-variance tradeoff better than the one achieved by the algorithm generating the predictors in the combination.[1] AdaBoost, short for adaptive boosting, is an algorithm that receives as input a training set of examples $S = \{(x_t, y_t)\}_{t=1}^{m}$. The boosting process proceeds in a sequence of consecutive rounds. At each round t, the algorithm defines a distribution D of weights. Initially, these weights are all equals. Then, the booster passes the sample S and the weights distribution to a weak learner. The weak learner returns a prediction h_t, whose error is at most $\frac{1}{2} - \gamma$.
After that, AdaBoost assign a weight to the weak learner's prediction h_t as follows: $w_t = \frac{1}{2} log(\frac{1}{\epsilon_t} - 1)$ where ϵ_t is the h_t error. At this point, AdaBoost updates the weights distribution D in such a way that examples bad classified by the weak learner will get a higher probability, well classified ones, instead, will get a lower probability. Intuitively, this force the weak learner to focus more on problematic example in the next round than samples of minority class. [2] The algorithm keeps repeating the training and weight-adjusting iteration until the training error is 0 or until it reaches a predefined number of weak classifiers.
The output of the algorithm is a weighted sum of all the weak predictions. The pseudocode of AdaBoost is presented in the following image.

AdaBoost

input:
training set $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
weak learner WL
number of rounds T
initialize $\mathbf{D}^{(1)} = (\frac{1}{m}, \ldots, \frac{1}{m})$.
for $t = 1, \ldots, T$:
invoke weak learner $h_t = \mathrm{WL}(\mathbf{D}^{(t)}, S)$
compute $\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} \mathbb{1}_{[y_i \neq h_t(\mathbf{x}_i)]}$
let $w_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$
update $D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(\mathbf{x}_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(\mathbf{x}_j))}$ for all $i = 1, \ldots, m$
output the hypothesis $h_s(\mathbf{x}) = \mathrm{sign}\left(\sum_{t=1}^{T} w_t h_t(\mathbf{x})\right)$.

Figure 1: Adaboost pseudocode - taken from the book "Understanding Machine Learning: from theory to algorithms". [2]

The algorithm just described is designed for binary classification, however, it can be generalized to deal with multiclass problems. In order to do so, it is possible to split the multi-class problem in binary classification problems. For each class, the problem is transformed into a binary classification using the one-vs-all approach. We consider one class as label 1 and the others as -1. For example, given classes Y=1,2,3, we will have three binary classifiers:

- **1:** 1(class 1) vs -1 (classes 2,3)
- **2:** 1(class 2) vs -1 (classes 1,3)
- **3:** 1(class 3) vs -1 (classes 1,2)

We fit 7 different binary Adaboost with the new labels for the dataset and the output of each binary classifier has the form $f(x) = sgn(g(x)w_t)$. After training all the binary classifiers, we can calculate the multiclass prediction as calculated as: $F(x) = argmax(g(x))$.

Decision Stump

A decision stump is a single level decision tree, it takes as input a training set S and a distribution of weights S, it finds a feature and a threshold and it uses them to classify all the samples. As said before, Adaboost assumes to receive from the weak learner a prediction whose error is at most $\frac{1}{2} - \gamma$. There are several ways to find a decision stump, in this project we find use two different types of decision stumps: the best one and a random one.

The **best decision stump** at each round of Adaboost returns the feature and the thresholds that minimize the weighted error F with respect to the data weight vector D. The algorithm uses a boolean variable: if the value is positive, samples greater than the threshold will be classified as 1 and those smaller than the threshold will be classified as -1. On the contrary, if the value of the boolean variable is negative, the sign of the inequality changes and the samples larger than the threshold will be classified as 1. The algorithm iterates over all the features in the dataset, and for each feature iterates over all the unique values to find a threshold that minimizes the weighted error F with respect to the data weight vector D. If the error is bigger than 0.5, then we change the value of the boolean variable. In the end it returns the feature j and the threshold θ that minimizes the weighted error.

The **random decision stump** works like the best decision stump, but instead of iterating over all features and all unique values for each of them, it randomly chooses a feature and a threshold. As said before, AdaBoost assumes to get a prediction with a weighted error at most $\frac{1}{2} - \gamma$. Here, we set $\gamma = 0.03$ and if the error is bigger than 0.47 we try another random couple feature-threshold.

Dataset

The Forest Cover Type dataset contains tree observation from different areas of the Roosvelt National Forest in Colorado. It contains 581012 samples and each sample is composed by the following attributes [3]:

- Elevation: Elevation in meters
- Aspect: Aspect in degrees azimuth
- Slope: Slope in degrees
- Horizontal Distance To Hydrology: Horz Distance to nearest surface water features
- Vertical Distance To Hydrology: Vert Distance to nearest surface water features
- Horizontal Distance to Roadways: Horz Distance to nearest roadway

- Hillshade 9am: Hillshade index at 9am, summer solstice
- Hillshade Noon: index at noon, summer soltice
- Hillshade 3pm: Hillshade index at 3pm, summer solstice
- Horizontal Distance to Fire Points: Horizontal Distance to nearest wildfire ignition points
- Wilderness Area (4 binary columns):Wilderness area designation, 0 (absence) or 1 (presence)
- Soil Type (40 binary columns): Soil Type designation, 0 (absence) or 1 (presence)
- Cover Type(7 types): Forest Cover Type designation

There aren't categorical values, the presence of 40 different columns "soil type" suggests that this feature has been already one-hot encoded, same for the feature "Wilderness Area". In the dataset there aren't null values, so there is no need to drop or deal with missing data.

The goal of the project is to predict the correct cover type using Adaboost and evaluate the algorithm at the increasing round of the number of weak learners used. I tried two different experiments: the first one uses best decision stumps, the second uses random stumps. For timing performance issues, I chose to take a 20k sample of the dataset.
The number of examples in the dataset for each class label is not balanced as we can see in the following barplot:

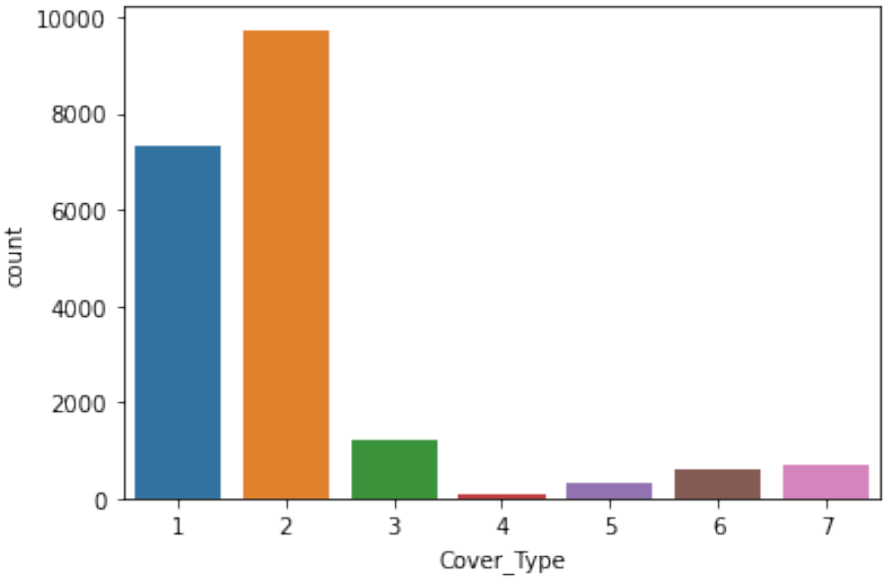


Figure 2: Distribution of Cover type

As we can see in figure 3, for each feature we create a barplot with respect to the cover type classes. It is interesting to see that some features are characteristic only of some classes. We can also see that some features do not appear at all in the subset found. So, we drop the feature "Soil Type15". At this we proceeded with the experiment. In the next section we illustrates the results obtained.

Figure 3: Distribution of each feature respect to the class

Results

The main goal of the project is to predict the cover type class and evaluate how the error changes as the number of rounds changes. We want to evaluate, therefore, the average error of the learning algorithm, to do so we use external cross validation.
The k-fold cross validation technique is designed to give an accurate estimate of the true error. The original dataset is divided into k subsets (or folds) of size m/k. For each fold, the algorithm is trained on the union of the other folds and the predictor is tested on the fold. After calculating the error of the individual predictors, we average all these errors and estimate the true error. [2]
Scikit-learn implements K-fold CV in the module KFold, we used it to perform external cross validation. We used 5 folds and we take the mean of training and test error.

First, we compare the results obtained on multiclass prediction with two different strategies. In the image 4 we can see on the left how the training and test error change as the number of best decision stumps used increases. As can be expected, as the number of decision stumps increases, the training error decreases. The test error always remains slightly higher than the training error and does not increase dramatically, we are not overfitting. On the right we see instead how the error changes using the strategy of random decision stump. Many more rounds are needed for the error to go down and it remains higher than the ones of the strategy using the best decision stump.

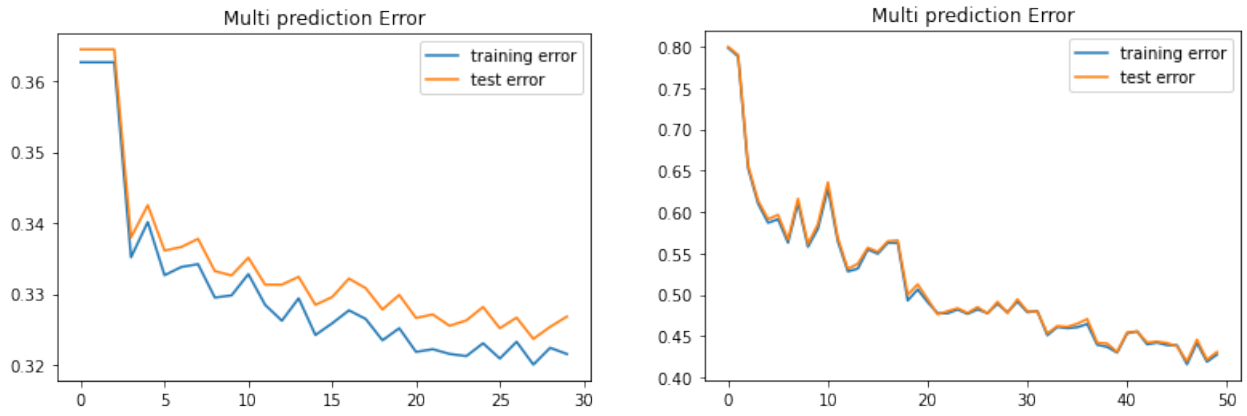


Figure 4: Training and test error on multi label prediction using best decision stumps (on the left) and random decision stumps(on the right)

In the following two pages are shown the results obtained on individual classes, in the first page we see the results obtained using adaboost that uses the best decision stump as weak learners, in the next page we see Adaboost trained with random decision stump as weak learners.

There are two considerations to make: the first is that despite the dataset is strongly unbalanced, the two worst performing classes are the ones that have more examples, with both best and random strategies. For this reason, trying to use a more balanced dataset would not change the results much. Secondly, we note that using best decision stumps as weak learners less rounds are required for training error to converge. Referring to the theory, it is an expected result. In fact, a number $T > \frac{ln(m)}{2\gamma^2}$ of boosting round is sufficient to bring the training error down to zero. With the random strategy, gamma is smaller and therefore a larger T is required.

Figure 5: Training and test error on binary models using AdaBoost with Best decision stumps as weak learners

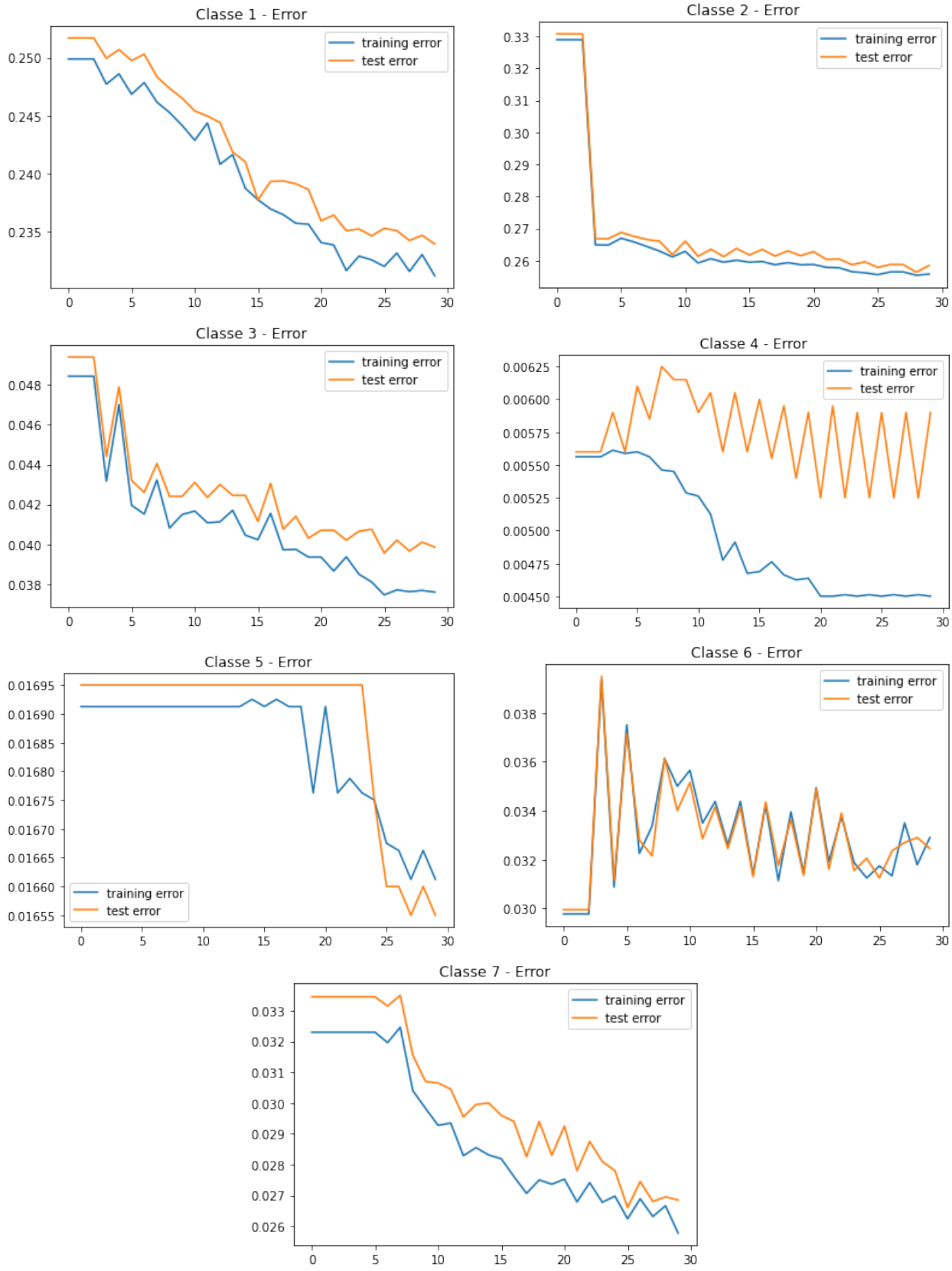
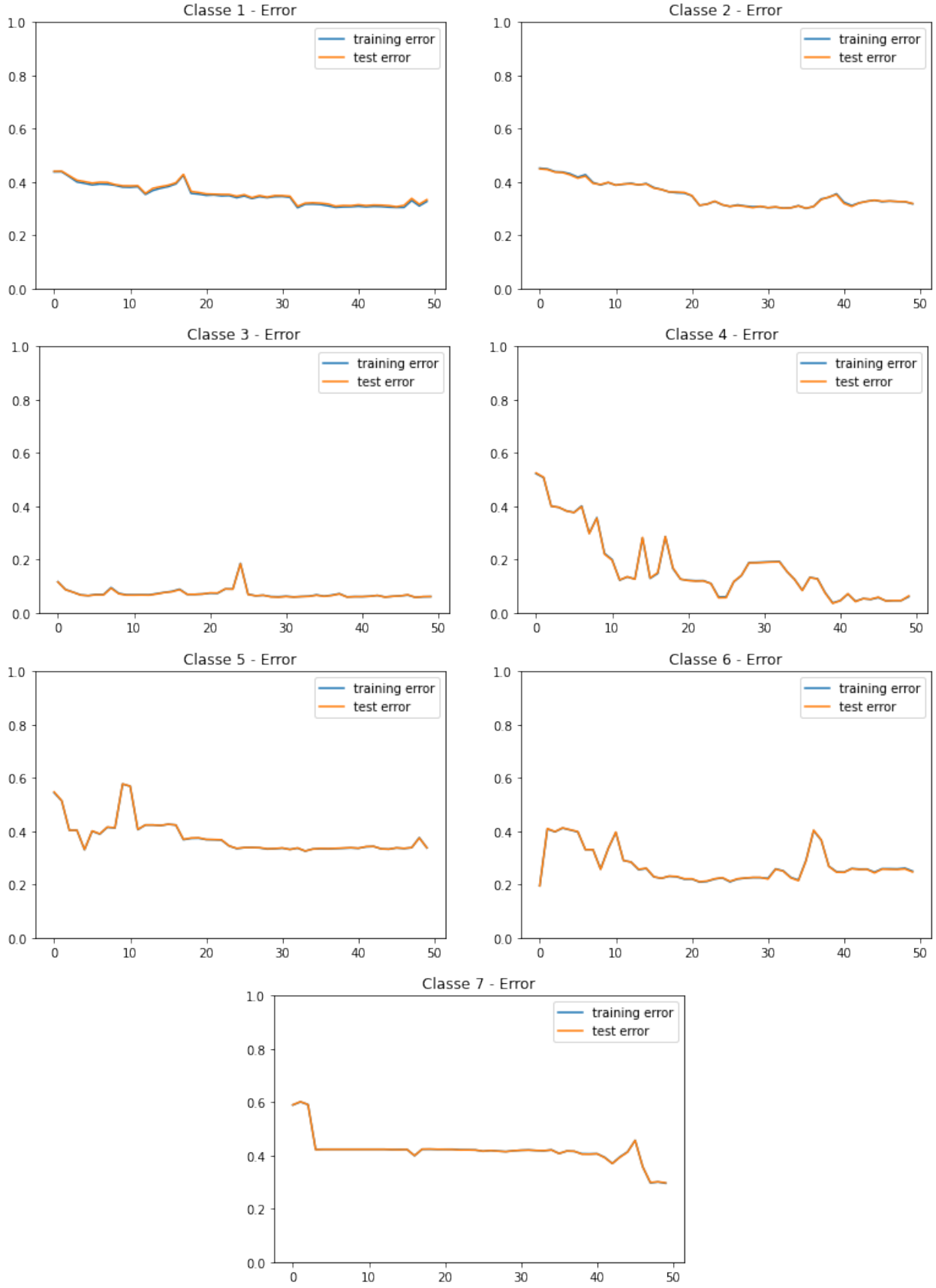


Figure 6: Training and test error on binary models using AdaBoost with Random decision stumps as weak learners



References

[1] Nicolò Cesa-Bianchi. Boosting and ensemble methods.

[2] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[3] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.