Natural Computing, Assignment 3

Dennis Verheijden - s4455770 Pauline Lauron - s1016609
Joost Besseling - s4796799

March 27, 2018

Combining, Bagging & Random Forests

1

(a)

- The probability that all three doctors give the correct answer is $0.8^3 = 0.512$.
- The probability that exactly 2 doctors make the right call is $0.8 * 0.8 * 0.2 + 0.8 * 0.2 * 0.8 + 0.2 * 0.8 * 0.8 = 0.384$. Therefore, the probability that *at least* two doctors make the right call is $0.512 + 0.384 = 0.896$.
- The probability that this group makes the right decision based on majority voting is $0.512 + 0.384 = 0.896$ since the majority is when there are at least two individuals.

(b)

The general formula is

$$P(\text{correct predictions} > c/2) \sum_{i=\lfloor n/2 \rfloor}^{n} p^i (1-p)^{n-i} \binom{n}{i}.$$

Using this formula, we find a probability of about 0.826.

(c)

If we use 10000 runs of the simulations, we get an approximately equal result of 0.826. Writing out more decimals gives us a difference of 0.0057. So our approximation is pretty good.

(d)

We decided to use a surfplot for the visualization. Here we can easily spot the differences when variables change relative to each other.

The surfplot can be found in figure 1. What we can observe from this plot is that the jury size matters for a low number of people, but that this effect has exponentially diminishing returns. The competence, as expected, has the highest effect on the probability of correctly making the right decision.

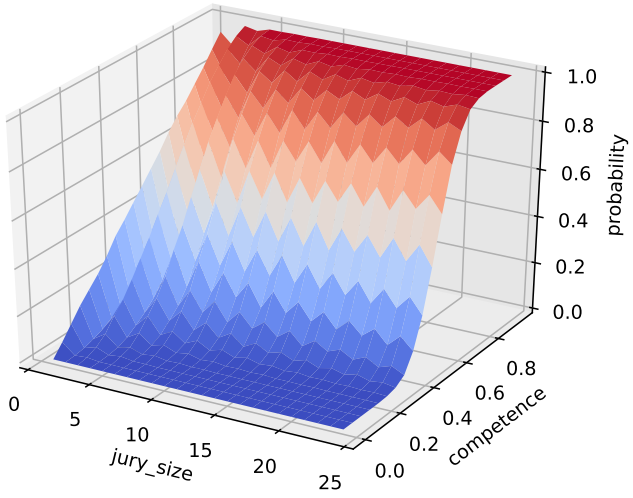

Figure 1: Surfplot of the probabilities as a function of the jury size c and competence p.

(e)

The probabilities for making the correct decision for the groups are:

- **radiologists**: 0.850
- **doctors**: 0.896
- **students**: 0.826

To reach the same probability for making the correct decision as the group of doctors, using only students. You would need 28 students. These would, collectively, have a probability of making the correct decision of 0.898.

2

The filled in table may be found below in table 1. Here we can see the values for different combinators. Colors denote the choice that the classifier will make. What we can note from this table is that the choices of these classifiers are in most cases the same.

$p_1(\omega\|x)$		$p_2(\omega\|x)$		$p_3(\omega\|x)$		Mean		Max		Min		Prod	
A	B	A	B	A	B	A	B	A	B	A	B	A	B
0.9	0.1	0.9	0.1	0.0	1.0	0.6	0.4	0.9	1.0	0.0	0.1	0.0	0.01
0.9	0.1	0.9	0.1	0.3	0.7	0.7	0.3	0.9	0.7	0.3	0.1	0.243	0.007
0.9	0.1	0.2	0.8	0.1	0.9	0.4	0.6	0.9	0.9	0.1	0.1	0.018	0.072
0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0

Table 1: Classifier Combination for different methods. Values indicated in orange denote the decision the combiner would make. Values indicated in red denote a tie, the choice here is based on the implementation.

3

The formula for the percentage of items that are not in a sample using Bootstrapping is:

$$p_{notPresent} = (1 - \frac{1}{N})^N$$

We also made simulations which approximate this function, as found in figure 2. Here we see that our simulations approximate our formula. Which adds evidence to its correctness.

Now that we have this implementation, we can also make simulations for different N. The results for this are shown in figure 3. Here we can see the comparisons for different N. We see that the probability of items that do not occur converge to 0.37 for larger N. We see that there probably is an exponential relation as the probability first rapidly increases to 0.25 for $N = 2$ and at $N = 5$ it is almost at convergence.

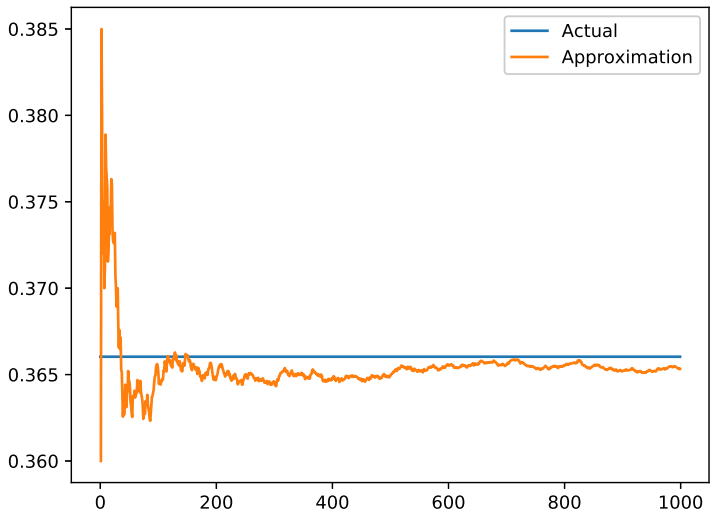

Figure 2: Comparison of our simulation and the exact value.

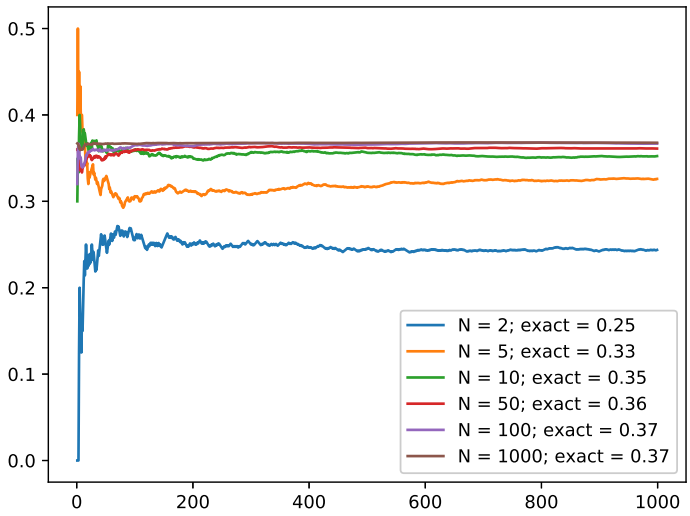

Figure 3: Comparison of different sample sizes for the percentage of left-out items.

4

The difference between bootstrapping and random subspaces is how the training data is made.

- For the bootstrapping, we selected randomly the data among all the training data.
- For the radom suspaces, we selected data among each training example.

5

For calculated the importance of variable, we can use the Mean Decrease Impurity. This compute how much each feature decreases the weighted impurity in a tree where the impurity is the measure based on which the optimal condition is chosen.

6

In order to implement a random forest, we used the Iris dataset, the sklearn library with the package RandomForestClassifier. We tested 3 parameters :

- n_estimators which is the number of tree. The more we increase this number, the more the score is stable and the performance seems better, but the code beging slower
- the max_depth which is the maximum depth of the tree. If we also increase this number, the performance seems better.
- the max_features which is the maximum number of features Random Forest is allowed to try in individual tree.

To test different combination of parameters, we use the Grid Search. We obtain a score around 0.93.

Boosting

1

2

3

4

5