Team Name: Liad & Yuval

ID #1: 300822954

ID #2: 311434047

**1. Algorithm Name**

RotBoost

**2. Reference**

[Zhang, Chun-Xia, and Jiang-She Zhang. "RotBoost: A technique for combining Rotation Forest and AdaBoost." Pattern recognition letters 29, no. 10 (2008): 1524-1536.](http://ce.sharif.ir/courses/88-89/1/ce717-1/assignments/files/assignDir2/+RotBoost.pdf)

**3. Motivation for the algorithm (or which problems it tries to solve?)**

AdaBoost and Rotation Forest are powerful learning algorithms with wide success on tabular data tasks. Combining the stochastic rotation procedure in Rotation Forest, which is based on PCA, in order to create a bagging of diverse collection of AdaBoost classifiers, may lead to a better learning algorithm.

From the paper: “In view of the fact that both AdaBoost and Rotation Forest are successful ensemble classifier generation techniques and they apply a given base learning algorithm to the permutated training sets to construct their ensemble members with the only difference lying in the ways to perturb the original training set, it is plausible that a combination of the two may achieve even lower prediction error than either of them”.

**4. Short Description:**

RotBoost is a tree ensemble algorithm. The algorithm is constructed by creating a bagging of AdaBoost classifiers. Each AdaBoost classifier is trained on a differently rotated input data (according to procedure which is based on PCA, see below for more details). The AdaBoost classifier’s base learner is a simple decision tree.

**5. Pseudo-Code**

**RotBoost - Building the ensemble**

Input:

* : a training set, . Where X is an matrix containing the input attribute values and Y is an N-dimensional column vector containing the class labels
* : number of attribute subsets (or Q: number of input attributes contained in each subset).
* I: a base learning algorithm
* : number of iterations for Rotation Forest
* : number of iterations for AdaBoost

Training Phase

1. **FOR**
2. Randomly split attribute set F into K subsets
3. **FOR**
4. Select the columns of X that correspond to the attributes in to compose a submatrix
5. Draw a bootstrap sample of size 75% from
6. Apply PCA to to obtain a matrix whose ith column consists of the coefficients of the ith principal component.
7. **END FOR**
8. Arrange the matrices into a block diagonal matrix
9. Construct the rotation matrix by rearranging the rows of so that they correspond to the original attributes in F.
10. Provide as the input of to build a classifier
11. Initialize the weight distribution over
12. **FOR**
13. According to the distribution , perform N extractions randomly from with replacement to compose a new set
14. Apply to to train a classifier , and then compute the error of as
15. **IF** , **then**
16. set and goto to step 14
17. **END IF**
18. **IF** , **then**
19. set to continue the following iterations
20. **END IF**
21. Choose
22. Update the distribution over as

Where is a normalization factor being chosen so that is a probability distribution over

1. **END FOR**
2. Let
3. **END FOR**

**RotBoost – classify an instance:**

Input:

* : a data point to be classified

Prediction:

1. The class label for x predicted by the final ensemble classifier as

**6. Algorithm Explanation:**

Generally, the algorithm performs Z iterations of rotation forest and for each iteration, performs T iterations of AdaBoost (for each one the input data is rotated differently, according to the procedure describe below).

The final model is a majority vote classifier of the AdaBoost classifiers.

The rotated input data for each AdaBoost algorithm is obtained by the following procedure:

* The feature set F is randomly split into K subsets
* Relevant columns from X that correspond to the attributes in the current k subset are selected. Denoted as
* Drawing a bootstrap sample of size 75% from above
* PCA is applied to each subset
* All principal components are retained
* PCA’s coefficients are arranged in a rotation matrix
* Applying the rotation matrix to the data features.

Denote as as the input of to build a classifier

After the data is created, AdaBoost algorithm is trained on it. The AdaBoost training contains some adjustments:

* Line 19: “If , the classifier is incorporated into the ensemble with instead of which will assign an infinite weight to the vote of the classifier for constructing the ensemble classifier ”.
* Line 16: in the original AdaBoost, “the iteration is carried out until T classifiers are generated or until a classifier achieves an error zero or an accuracy below 0.5, in which cases the weight updating rule fails and the algorithm stop. In order to prevent early stopping when a classifier has , one common variant of AdaBoost is to reset the weight to continue the boosting process”.

Note: “These modifications allow the boosting process to always generate the specified number of base classifiers and, in general, increase the prediction accuracy of AdaBoost”.

Furthermore, in the paper they mentioned that this allow them to “make the comparisons with other ensemble methods that always produce the desired number of base classifiers much fairer”.

Reference: <http://www.ehu.eus/ccwintco/uploads/b/ba/RotationForest-intro.pdf>

**7. Illustration**

**Figure that can explain the essence of how the algorithm works**

Z iterations of Rotation Forest. Namely, follows similar steps to those in Rotation Forest in order to compute the rotation matrix so it can apply it to the data features and generate a rotated training data for classifier

…

Updated Weighted Training Dataset, D3

Updated Weighted Training Dataset, D2

Build T AdaBoost models, each is given with a different training dataset

Model-2

(iteration 2)

Model-2

(iteration 2)

Model-1

(iteration 1)

Prediction on weighted data, D3

Prediction on weighted data, D2

Prediction on given Training data, denote as D1

N Instances

Training Dataset

Model-1

(iteration 1)

Model-2

(iteration 2)

Prediction on given Training data, denote as D1

Prediction on weighted data, D2

Model-2

(iteration 2)

Prediction on weighted data, D3

Updated Weighted Training Dataset, D2

Updated Weighted Training Dataset, D3

Majority Voting

…

Model-1

(iteration 1)

Model-2

(iteration 2)

Prediction on given Training data, denote as D1

Prediction on weighted data, D2

Model-2

(iteration 2)

Prediction on weighted data, D3

Updated Weighted Training Dataset, D2

Updated Weighted Training Dataset, D3

…

* **Demonstration how the algorithm works:**

In this section we illustrate RotBoost algorithm using the following settings:

Z=3, T=3, K=3, AdaBoost learning rate=1, random\_state=0, and we use a decision tree stump as base inducer for AdaBoost.

[schlvote dataset](https://openml2.win.tue.nl/d/848) (Table 1) is used as the training set, which have 38 samples and 4 features.

We present for each classifiers the rotated training data and in each we presents AdaBoost’s inner estimators and their weights and errors and show samples reweighting updates as depict in line 23, in other words, the weights assigned to each training instance.

**:**

Rotated training data for classifier :

X: [[ 0.6639831 0.11407545 0.64980245 0.6223468 0.8053924 ]

[ 0.78700113 0.19758801 -0.6711692 -0.49230936 -0.38774225]

[ 0.76671135 0.19484416 0.33093786 -0.38578603 1.7238885 ]

[-0.7745744 1.1258557 0.30909526 -0.2813006 -0.15466432]

[-1.2128639 1.3950272 -0.4833716 -0.2825947 -0.5086708 ]

[ 0.49011073 0.3453405 -0.6922144 -0.25078285 1.1784276 ]

[ 0.42556542 0.42092082 -1.0188317 -0.50291353 -0.02966993]

[-1.1065794 -1.6364923 0.2213892 0.18778239 0.6718131 ]

[ 0.8801025 0.09678931 -0.14535154 -0.18625207 0.63184685]

[-0.56119174 -1.9824315 -0.24576348 0.26646414 0.40561515]

[ 1.1448629 -0.01365509 -0.62219244 -0.55107754 -0.20267624]

[-0.07029676 0.7420864 0.016586 -0.6209985 -0.82011336]

[ 0.10117137 0.59033144 0.15212251 -0.29488498 -0.6143818 ]

[ 0.4165459 0.3800535 -0.12463593 -0.17763332 -0.81302714]

[-1.9045867 -1.078422 0.51968896 -0.29111987 0.09118102]

[-0.53025985 -1.944309 0.65102816 -0.13384525 1.1763324 ]

[ 1.6722554 -0.44147748 -0.08073711 0.17901579 1.3204528 ]

[ 0.224788 0.5290646 0.81092644 -0.39722344 1.8988463 ]

[-0.4951891 0.9659398 0.5101425 -0.3622529 -0.91982454]

[ 0.503976 0.3698162 -0.6598081 -0.4820061 -0.51812893]

[ 0.88567954 0.14930138 -0.7465496 -0.578351 -0.28071478]

[ 0.509553 0.42232826 -0.85204196 -0.87410504 -0.7730138 ]

[ 0.3691987 0.46883568 -0.91352224 -0.5962965 -0.57107127]

[ 1.1115125 0.09915785 1.1256679 -1.1984105 2.1551461 ]

[ 0.60101646 0.18431558 -0.7086514 0.40085915 -0.5922685 ]

[ 0.36673525 0.40579146 -0.25463173 -0.14376284 -0.530545 ]

[-0.7872285 -1.7888436 -0.32567927 -0.11813342 -0.52766454]

[ 0.53288835 -0.01349379 0.05430075 2.080584 -0.20924662]

[-1.3643321 -1.8352618 0.40884247 2.6900265 -0.4092451 ]

[-2.0290968 -1.371019 5.544002 2.295623 -0.25450727]

[ 1.7915734 -0.7023351 -1.1719798 1.4942875 0.43911698]

[ 0.7524569 -0.18520145 -0.35013974 2.3396304 0.2102739 ]

[-3.2388873 2.6809514 -1.379507 -0.58061725 -0.54023284]

[ 0.7053092 0.33116952 0.11483276 -1.0772568 0.5100705 ]

[ 0.10859022 0.6366133 1.4152279 -0.6512338 -2.1664557 ]

[-0.07501099 0.75803304 -0.30290094 -0.7125019 -0.05019365]

[ 0.41748875 0.37686417 -0.7973568 -0.15933265 -0.46855912]

[-2.0789788 -0.98815364 -0.2875572 -0.17363721 -0.87578636]]

Iteration 1:

Estimator weight: 0.66087792

Estimator error: 0.21052632

Samples weights:

[0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

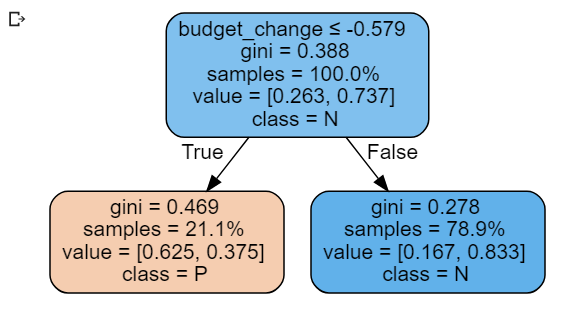
0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579]



Iteration 2:

Estimator weight: 0.58290549

Estimator error: 0.23761301

Samples weights:

[0.02198192 0.02198192 0.02198192 0.02198192 0.02198192 0.02198192

0.02198192 0.02198192 0.0425678 0.02198192 0.02198192 0.0425678

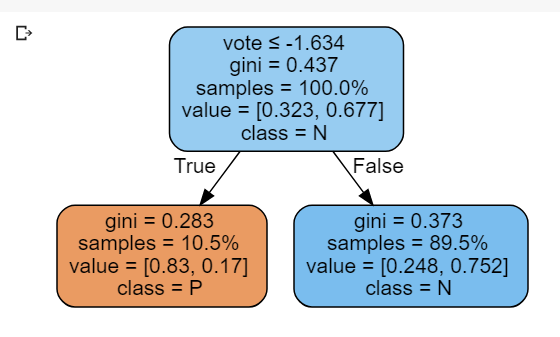
0.02198192 0.02198192 0.0425678 0.02198192 0.02198192 0.02198192

0.02198192 0.02198192 0.02198192 0.02198192 0.02198192 0.0425678

0.02198192 0.02198192 0.0425678 0.02198192 0.02198192 0.02198192

0.02198192 0.02198192 0.02198192 0.0425678 0.02198192 0.02198192

0.0425678 0.0425678 ]



Iteration 3:

Estimator weight: 0.44523481

Estimator error: 0.29101292

Samples weights:

[0.01850318 0.01850318 0.01850318 0.01850318 0.01850318 0.01850318

0.01850318 0.01850318 0.0641822 0.01850318 0.01850318 0.03583125

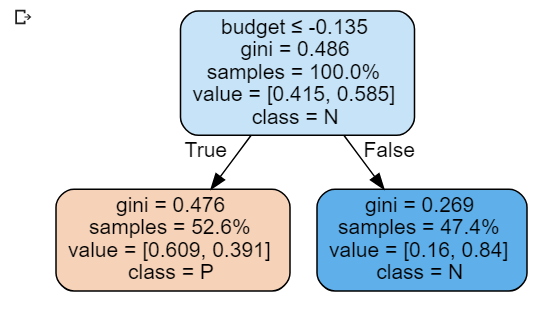
0.01850318 0.01850318 0.03583125 0.01850318 0.01850318 0.01850318

0.01850318 0.01850318 0.01850318 0.03314354 0.03314354 0.03583125

0.01850318 0.01850318 0.0641822 0.01850318 0.01850318 0.03314354

0.01850318 0.01850318 0.01850318 0.03583125 0.03314354 0.03314354

0.0641822 0.03583125]



:

Rotated training data for classifier :

X: [[-0.80645716 -0.09321459 0.26498663 0.9134202 0.5903269 ]

[ 0.12105355 -0.69342893 -0.89256036 0.03206674 -0.45651108]

[ 0.033532 -0.63679165 0.53839743 1.7687154 0.17634986]

[ 0.14683795 -0.7101146 0.9475731 -0.7978217 0.07660971]

[ 0.20530666 -0.7479511 0.9573026 -1.3875663 -0.7389474 ]

[-0.04454741 -0.58626455 0.2408117 1.1199849 -0.7950888 ]

[ 0.17734337 -0.7298554 -0.45256072 0.08581925 -0.9347645 ]

[ 0.70847905 1.8481224 0.5364892 0.47598058 0.07972174]

[-0.15022698 -0.5178768 -0.27416125 0.930146 -0.0888565 ]

[ 0.57047826 1.937426 -0.23161101 0.6067477 -0.20011435]

[ 0.12395885 -0.695309 -1.1144038 0.42259815 -0.35139588]

[ 0.34221804 -0.8365497 -0.16005296 -0.8755277 0.07099158]

[ 0.04370049 -0.6433719 -0.15680136 -0.60251063 0.20699616]

[-0.09684245 -0.5524232 -0.6235517 -0.56628007 0.04485879]

[ 1.2183558 1.518169 1.0159625 -0.50776243 0.27268052]

[ 0.90531176 1.7207472 0.37124315 1.2871461 0.55847615]

[-0.5631399 -0.25067097 -0.60117364 2.0073445 0.05193133]

[ 0.11415353 -0.6889637 1.2602042 1.5628604 0.48279917]

[ 0.17879601 -0.73079544 0.3201663 -1.2399269 0.45809022]

[ 0.14938001 -0.7117596 -0.69721067 -0.26178485 -0.4932947 ]

[ 0.18097498 -0.7322055 -0.94958913 0.18721828 -0.52124745]

[ 0.480582 -0.92608833 -0.9093951 -0.45736876 -0.6330925 ]

[ 0.26377544 -0.78578764 -0.6718538 -0.39328527 -0.7610708 ]

[ 0.67632526 -1.0527585 0.61913025 2.3882926 0.96968186]

[-0.6107139 -0.21988472 -0.79801434 -0.2896177 -0.5165498 ]

[-0.11899521 -0.5380876 -0.45566761 -0.36880547 -0.13709807]

[ 0.92564875 1.7075868 -0.5682042 -0.29948914 -0.18142425]

[-2.0237696 0.6945374 -0.24903688 -0.06617965 0.12364654]

[-1.3760595 3.197077 0.35617626 -0.6675339 0.336389 ]

[-0.955157 2.924701 2.3653271 -0.9012785 5.118033 ]

[-1.6922044 0.4799736 -1.4101559 1.3011621 -0.8525089 ]

[-2.271808 0.85504884 -0.31691214 0.41048276 -0.28288445]

[ 0.7228098 -1.0828397 2.5757391 -2.7466774 -2.0823624 ]

[ 0.6269355 -1.0207973 -0.15347347 0.7481268 0.15050602]

[ 0.34439695 -0.83795977 -0.6983336 -1.8501678 1.6751113 ]

[ 0.42029744 -0.8870768 0.17603576 -0.24447474 -0.35756987]

[-0.11245833 -0.5423178 -0.6100432 -0.29038233 -0.65863925]

[ 1.141729 1.567756 0.4492212 -1.4336715 -0.39977896]]

Iteration 1:

Estimator weight: 0.66087792

Estimator error: 0.21052632

Samples weights:

[0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

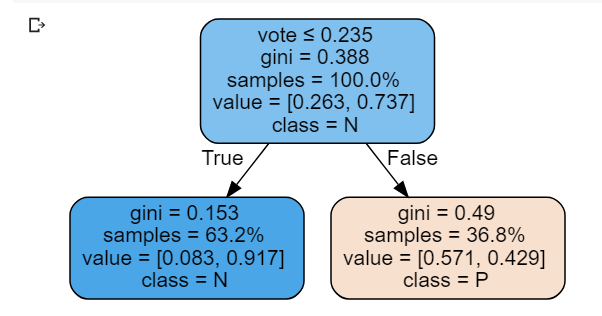
0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579]



Iteration 2:

Estimator weight: 0.70877739

Estimator error: 0.1950452

Samples weights:

[0.02198192 0.02198192 0.02198192 0.02198192 0.02198192 0.02198192

0.02198192 0.0425678 0.0425678 0.0425678 0.02198192 0.0425678

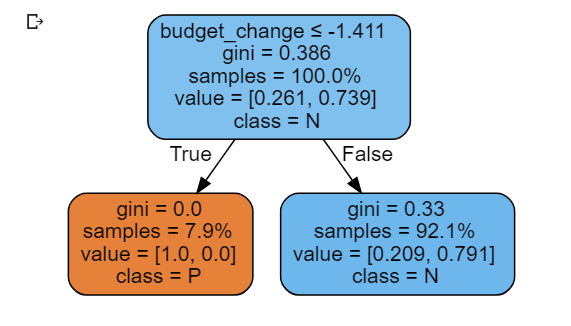
0.02198192 0.02198192 0.02198192 0.0425678 0.02198192 0.02198192

0.02198192 0.02198192 0.02198192 0.02198192 0.02198192 0.0425678

0.02198192 0.02198192 0.02198192 0.02198192 0.02198192 0.02198192

0.02198192 0.02198192 0.02198192 0.0425678 0.02198192 0.02198192

0.0425678 0.02198192]



Iteration 3:

Estimator weight: 0.44232243

Estimator error: 0.29221617

Samples weights:

[0.01830011 0.01830011 0.01830011 0.01830011 0.01830011 0.01830011

0.01830011 0.03543802 0.07199255 0.03543802 0.01830011 0.03543802

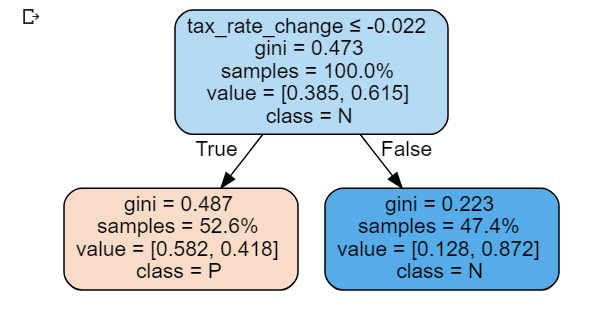
0.01830011 0.01830011 0.03717679 0.03543802 0.01830011 0.01830011

0.01830011 0.01830011 0.01830011 0.03717679 0.03717679 0.03543802

0.01830011 0.01830011 0.03717679 0.01830011 0.01830011 0.01830011

0.01830011 0.01830011 0.01830011 0.03543802 0.01830011 0.03717679

0.07199255 0.01830011]



:

Rotated training data for classifier :

X: [[-0.21036224 0.7578787 0.98434484 0.01246204 -0.5695437 ]

[-0.7398578 -0.02380226 -0.5238029 -0.7959566 0.2138221 ]

[-0.2746099 0.04409138 0.9506461 -0.71135825 -1.5391037 ]

[-0.3812986 -1.1031898 0.4167514 0.6804057 0.25895426]

[-0.67988926 -1.4539119 -0.38895 1.0566324 0.38148293]

[-0.6647774 -0.07003932 -0.1137988 -0.38831478 -1.2960093 ]

[-0.84380466 -0.3205533 -0.73670554 -0.49171242 -0.2254415 ]

[ 1.9223691 0.25206915 -0.30297783 0.35262915 -0.6804761 ]

[-0.50039446 0.292062 0.22565138 -0.68313354 -0.6210564 ]

[ 1.7433221 0.749327 -0.7936799 -0.06695067 -0.5530134 ]

[-0.7128064 0.21533938 -0.42932647 -1.1408532 0.04953399]

[-0.51968735 -0.8093981 -0.0270691 -0.13957407 0.8177531 ]

[-0.46096987 -0.4151759 0.15048358 -0.08152481 0.6573693 ]

[-0.5692008 -0.07099562 -0.15143597 -0.27892232 0.7721992 ]

[ 1.9971528 -0.7626546 -0.19516282 0.7372597 -0.04102344]

[ 2.1007204 0.45802963 0.21942617 -0.34588352 -1.0471017 ]

[-0.44128975 1.2124194 0.47198617 -1.134354 -1.2646239 ]

[-0.09583573 -0.39732417 1.4278463 -0.2523732 -1.5754229 ]

[-0.35059145 -0.9441733 0.38706008 0.38903612 1.0489326 ]

[-0.742714 -0.24148852 -0.5493423 -0.5459874 0.3421551 ]

[-0.76084715 -0.01298451 -0.5619174 -0.9350795 0.09035628]

[-0.82404953 -0.5465351 -0.79106283 -0.79793334 0.534186 ]

[-0.835128 -0.43923715 -0.79075694 -0.5020837 0.32338336]

[ 0.02885034 -0.3222308 1.7795236 -1.5201908 -1.735199 ]

[-0.76387405 0.53281176 -0.613327 -0.07297875 0.39995265]

[-0.60023916 -0.08400328 -0.19035728 -0.214723 0.4652516 ]

[ 1.6659214 0.26540238 -1.1206931 -0.11492047 0.3213259 ]

[-0.47418067 1.804315 0.17392147 1.0443951 0.24146096]

[ 1.9316308 2.0215168 -0.43132216 2.1515803 0.40914205]

[ 3.7537832 1.1798011 4.203868 2.4748492 1.6695335 ]

[-0.873206 2.3458066 -0.74532264 -0.40795478 -0.7176578 ]

[-0.594947 2.1839771 -0.07293779 1.0187918 -0.27239275]

[-0.99811286 -3.3056662 -1.199084 2.611687 0.16591027]

[-0.41490003 -0.5507002 0.42501387 -1.0943851 -0.43271986]

[-0.09654868 -0.6905341 0.85522205 -0.31252107 2.4944623 ]

[-0.5919827 -0.88538605 -0.10200557 -0.19319512 -0.00929362]

[-0.7886916 -0.05579803 -0.6587916 -0.2681981 0.25681034]

[ 1.6610471 -0.80906504 -1.181915 0.9613338 0.6661016 ]]

Iteration 1:

Estimator weight: 0.83698822

Estimator error: 0.15789474

Samples weights:

[0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

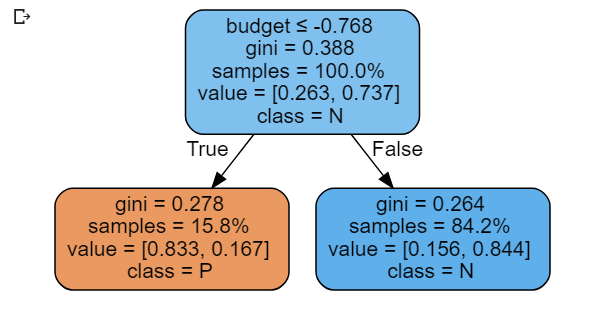
0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579 0.02631579 0.02631579 0.02631579 0.02631579

0.02631579 0.02631579]



Iteration 2:

Estimator weight: 0.5997438

Estimator error: 0.23156638

Samples weights:

[0.0218072 0.0218072 0.0218072 0.0218072 0.0218072 0.0218072

0.0218072 0.0218072 0.05036158 0.05036158 0.0218072 0.0218072

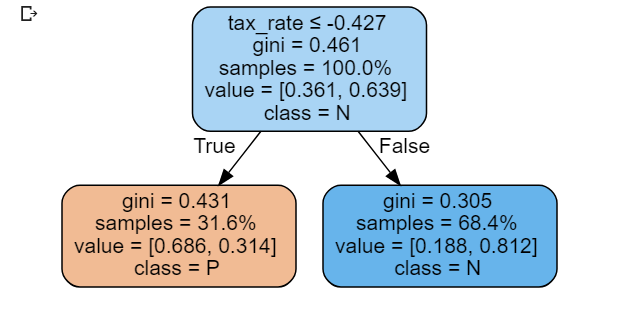
0.0218072 0.0218072 0.05036158 0.0218072 0.0218072 0.0218072

0.0218072 0.0218072 0.0218072 0.0218072 0.0218072 0.0218072

0.0218072 0.0218072 0.0218072 0.0218072 0.0218072 0.0218072

0.0218072 0.0218072 0.0218072 0.0218072 0.05036158 0.05036158

0.05036158 0.0218072]



Iteration 3:

Estimator weight: 0.22839841

Estimator error: 0.38774598

Samples weights:

[0.01832127 0.01832127 0.01832127 0.03337498 0.03337498 0.01832127

0.01832127 0.01832127 0.07707621 0.04231116 0.01832127 0.03337498

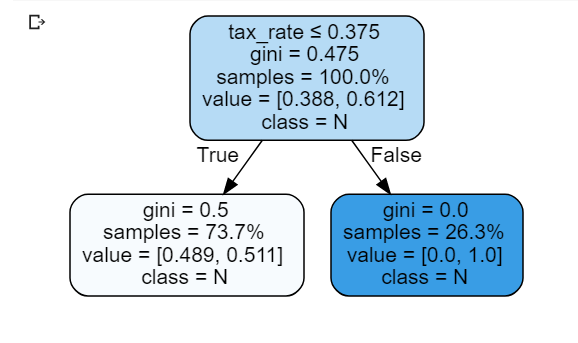
0.01832127 0.01832127 0.04231116 0.01832127 0.01832127 0.01832127

0.03337498 0.01832127 0.01832127 0.01832127 0.01832127 0.01832127

0.01832127 0.01832127 0.03337498 0.01832127 0.01832127 0.01832127

0.01832127 0.01832127 0.01832127 0.03337498 0.04231116 0.04231116

0.07707621 0.01832127]



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 1: Initial dataset |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | vote | tax\_rate | budget | budget\_change | tax\_rate\_change | binaryClass | | 1 | 47.18 | 47532280 | 4 | 6.3 | P | | 1 | 21.64 | 11637417 | 3.3 | 4.1 | P | | 1 | 24.05 | 59856099 | 3.4 | 2.5 | P | | 1 | 20.93 | 26926416 | 9.4 | 7.5 | P | | 1 | 19.32 | 11686005 | 11.1 | 5.2 | P | | 1 | 26.2 | 38641360 | 4.5 | -0.23 | P | | 1 | 20.09 | 13882048 | 4.7 | 1.7 | P | | 0 | 42.16 | 40298680 | 5.9 | 4.9 | P | | 1 | 29.11 | 35400375 | 3 | 3.5 | N | | 0 | 45.96 | 30314067 | 3.8 | 3.7 | P | | 1 | 21.56 | 15418131 | 1.9 | 3.8 | P | | 1 | 15.55 | 11997930 | 6.6 | 8.1 | P | | 1 | 23.77 | 17130165 | 6 | 8.1 | P | | 1 | 27.64 | 10502721 | 4.8 | 7.5 | P | | 0 | 28.12 | 33616668 | 8.9 | 7.7 | N | | 0 | 36.74 | 53999209 | 3.6 | 5.3 | P | | 1 | 40.48 | 48119900 | 0 | 1.9 | P | | 1 | 21.83 | 68400604 | 5.5 | 4 | P | | 1 | 20.05 | 15919780 | 8.3 | 10.4 | P | | 1 | 20.86 | 9499412 | 4.4 | 4.5 | P | | 1 | 19.99 | 12635368 | 2.9 | 3.5 | P | | 1 | 11.74 | 2862323 | 4.3 | 4.4 | N | | 1 | 17.71 | 5670720 | 4.9 | 3.6 | N | | 1 | 6.35 | 76466312 | 1.9 | 4.6 | P | | 1 | 41.79 | 7649906 | 4.2 | 4.5 | P | | 1 | 28.25 | 13926965 | 5 | 6.2 | P | | 0 | 36.18 | 13162830 | 4.6 | 5.9 | N | | 1 | 80.7 | 23056816 | 4.8 | 6.6 | P | | 0 | 99.56 | 23640880 | 7.4 | 8.6 | P | | 0 | 87.97 | 85184456 | 9.9 | 29.3 | P | | 1 | 71.57 | 20281917 | -0.2 | -0.2 | P | | 1 | 87.53 | 25719696 | 4 | 3.8 | P | | 1 | 5.07 | 866746 | 18.9 | 1.6 | N | | 1 | 7.71 | 36263806 | 3.5 | 4.9 | P | | 1 | 15.49 | 4606127 | 5.9 | 17.5 | N | | 1 | 13.4 | 21729978 | 6.6 | 4.7 | N | | 1 | 28.07 | 8785345 | 4.8 | 3.8 | N | | 0 | 30.23 | 7543792 | 9.6 | 7 | N | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

**8. Strengths**

1. In contrast to AdaBoost, RotBoost can be parallelized, “RotBoost has a potential advantage over AdaBoost of suiting parallel execution”
2. “RotBoost is found to perform much better than Bagging and MultiBoost”
3. “RotBoost can create ensemble classifiers with significantly lower prediction error than either Rotation Forest or AdaBoost more often than the reverse”
4. “Through employing the bias and variance decompositions of error to gain more insight of the considered classification methods, RotBoost is seen to simultaneously reduce the bias and variance terms of a single tree and the decrement achieved by it is much greater than that done by the other ensemble methods, which leads RotBoost to perform best among the considered classification procedures”
5. Due to RotBoost reliance on AdaBoost, it absorbs AdaBoost advantages such as the ability to adopt other algorithm such as neural networks as base learning algorithm.
6. The modification to AdaBoost as depicted above “allow the boosting process to always generate the specified number of base classifiers and, in general, increase the prediction accuracy of AdaBoost”.

**9. Drawbacks**

1. Need to tune the number of Rotation-Forest & AdaBoost iterations, parameters Z, T. Also, can require more hyper-parameter tuning such as AdaBoost’s learning-rate, number of attribute subsets etc.
2. Applying PCA can be computationally intensive (also requires normalizing the data).
3. Multi-class classification - although AdaBoost was also proposed to be used in the multi-class case (Freund & Schapire 1997), the paper [Multi-class AdaBoost](https://web.stanford.edu/~hastie/Papers/samme.pdf), depict a problem that practically, restricted reducing the multi-class classification problem to multiple two-class problems. RotBoost uses the Freund & Schapire (1997) AdaBoost algorithm, thus requires wrapping RotBoost with One-Vs-Rest classifier (or ovo), which also has impact on training-time and inference-time.
4. The current implementation suggested by the paper does not support Regression problems. However, supporting regression problems requires simple extension.
5. We can state that RotBoost algorithm is less easy to grasp comparing to other ensemble methods such as random forest for instance.

**10. Experimental Results**

We have conducted a nested-cross-validation (10 outer folds + 3 inner folds) with a Bayesian hyper-parameter tuning in order to make a comparison between RotBoost and Rotation Forest, on over 150 datasets.

Remarks:

* We choose ROC-AUC metric for the inner 3 folds cross-validation.
* We choose to use “macro” aggregation when calculating relevant metrics.
* Datasets - for details about dataset metadata see “dataset-metadata.csv” file.

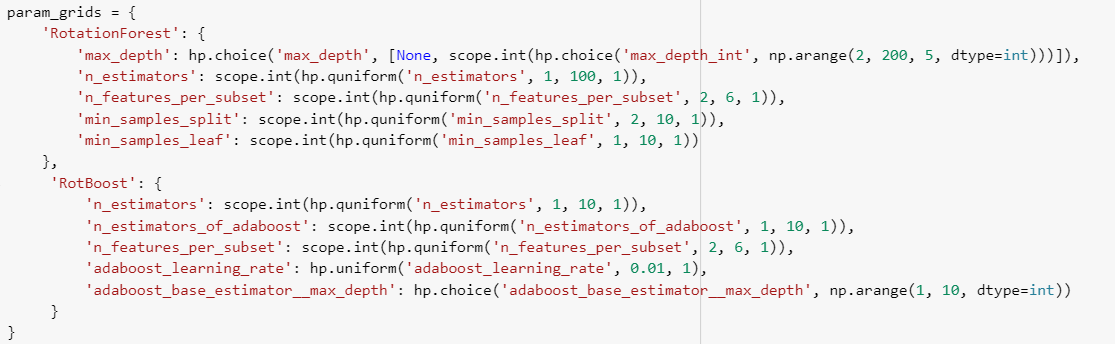
Chosen hyper-parameters explanation:

Similar to the paper [Chun-Xia Zhang and Jiang-She Zhang. Rotboost], in order to provide a fair comparison, for Rotation Forest, up to 100 trees can be trained to constitute the corresponding ensemble classifiers. With respect to RotBoost, the number of iterations for Rotation-Forest and AdaBoost can be up to 10, so that an ensemble classifier created by it can consists of 100 trees as well.

As for the other parameters, we set the search space of hyper-parameters, such as K, AdaBoost’s learning rate, decision tree’s attributes, etc. For instance, RotBoost AdaBoost’s base learner can be a stump or reach up to 10 depth, learning rate starting from 0.01, since up to 10 AdaBoost estimators can be constructed, etc.

Then, during the inner fold cross-validation Bayesian Search is applied, performing 50 trials per model on the given parameters grid.

The hyper-parameter search space:



**RotBoost Vs Rotation-Forest results:**

After running the experiment, the following results have emerged:

For AUC, Rotation-Forest is outperformed RotBoost on 93 out of 150 datasets.

Regarding training-time RotBoost is outperformed Rotation-Forest on 102 out of 150 datasets.

A statistical hypothesis test has been conducted, using Mann-Whitney rank test (two-side-tail test), and setting a significance cutoff alpha of 0.05.

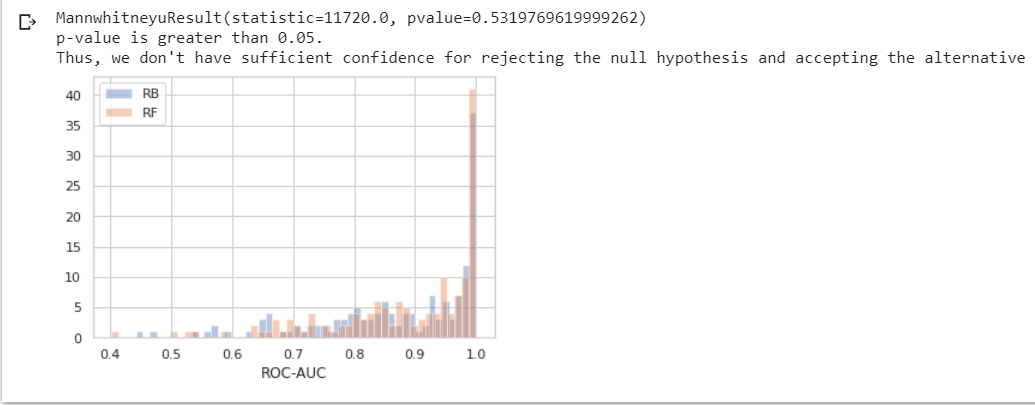
Rotation-Forest AUC scores (an average of all outer folds per dataset) were compared to RotBoost AUC scores, based on experimental results gathered above.

In more specific terms, test the hypothesis that the mean of population "Rotation-Forest ROC-AUC" (RF) is the same as "RotBoost ROC-AUC" (RB)

The hypotheses can be stated as follows:

* Null hypothesis H₀:
* Alternative hypothesis Hₐ:  (two-tail test in both direction)

**The test concludes that we don’t have sufficient confidence for rejecting the null hypothesis (p-value=0.53). In other words, the two algorithms perform the same on the given datasets.**



**Meta learning model:**

We use the meta features csv file and for each dataset we add the following label: 1 in case RotBoost has better AUC average score for a specific dataset and 0 for the opposite.

Note: basic pre-processing has been performed to handle missing values.

The meta learning model we trained is xgboost, with default parameters and metrics scores were gathered by performing Leave-one-out protocol.

The following results have emerged:

Accuracy 0.58

TPR 0.525

FPR 0.474

Precision 0.425

AUC 0.548

PR-Curve 0.439

Training Time 0.166

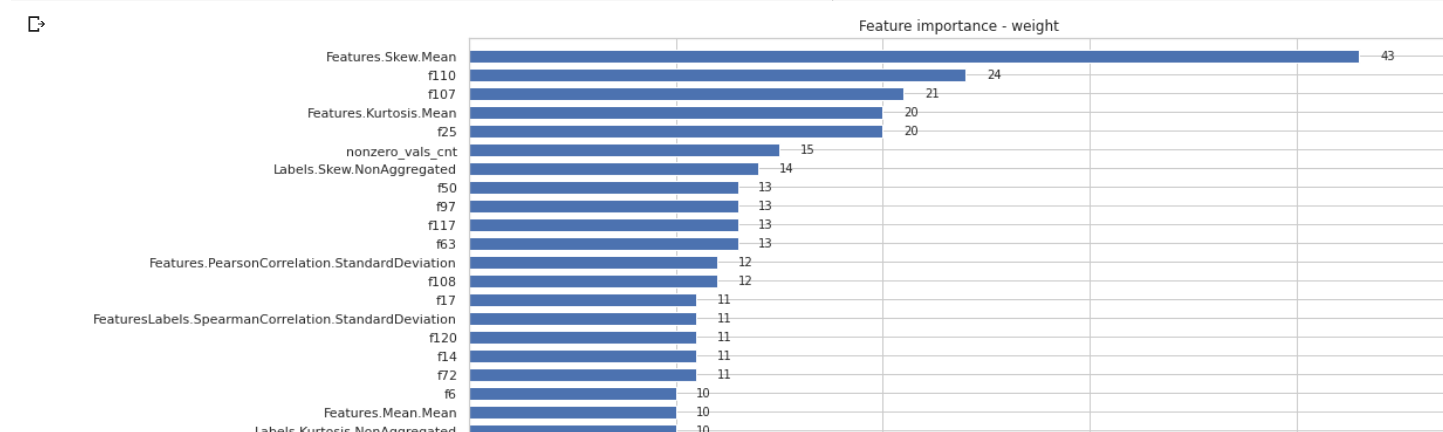
Inference Time 0.025

Namely, based on the provided features, we were not able to learn to classify which algorithm (RotBoost or Rotation Forest) is better for a given dataset.

We try to explain this result in the next section.

The Xgboost’s features importance (top-20):

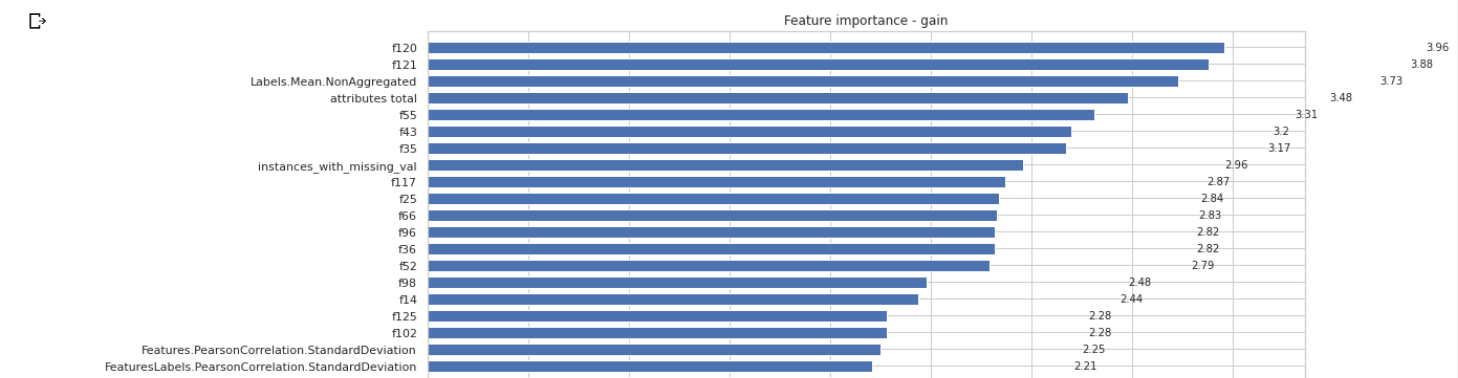
1. **Weight.**The number of times a feature is used to split the data across all trees.



1. **Cover.**The number of times a feature is used to split the data across all trees weighted by the number of training data points that go through those splits.

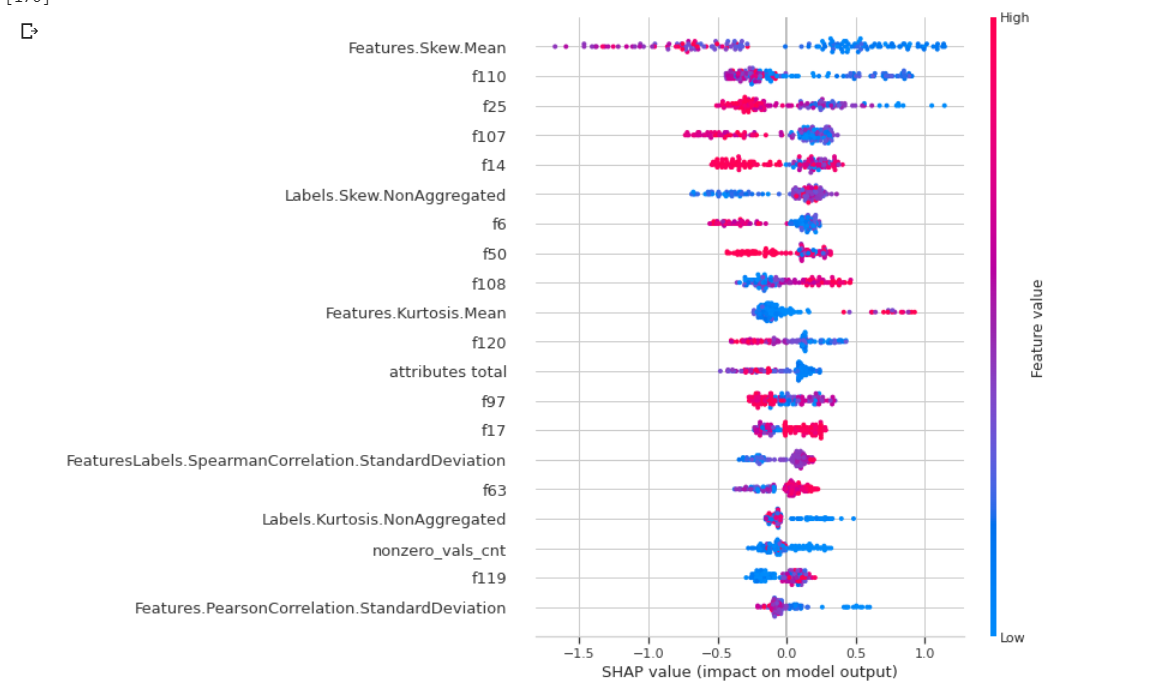


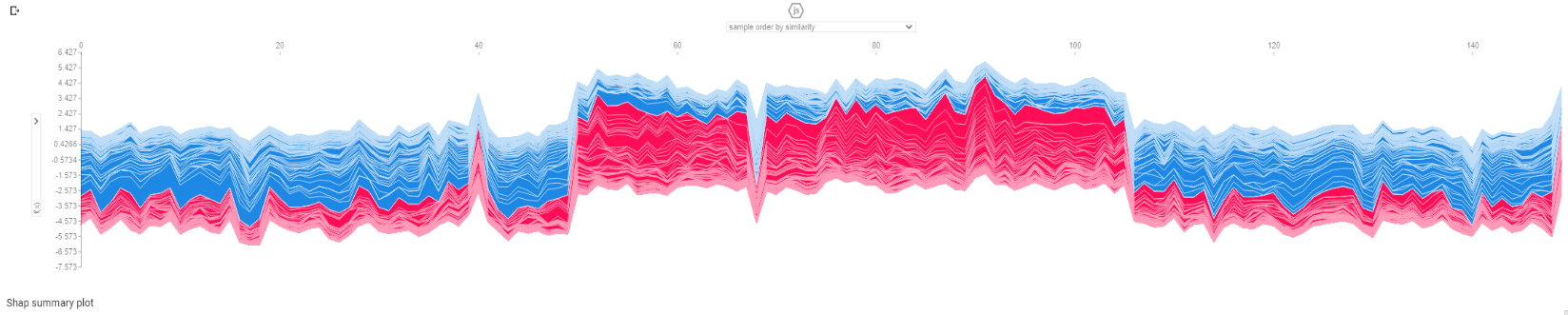
1. **Gain.** The average training loss reduction gained when using a feature for splitting.



**Shap**

Summary plot:



Visualization the training set predictions:

Note: in the notebook we also display dependence contribution plots and a specific training prediction explanation.

**11. Conclusions**

In this work we tried to evaluate the performance of RotBoost which is a novel ensemble classifier proposed by the paper “RotBoost: A technique for combining Rotation Forest and AdaBoost” by Chun-Xia Zhang and Jiang-She Zhang.

In order to do that, we implemented RotBoost algorithm according to the paper and compare it to Rotation-Forest (as implemented in <https://github.com/joshloyal/RotationForest>) on over 150 datasets. We used nested-cross-validation (10 outer folds + 3 inner folds) with a Bayesian hyper-parameter tuning.

After that, we tried to use datasets’ metadata, in order to learn meta classifier to predict which algorithm (RotBoost or Rotation-Forest) to use for a given dataset.

We found that neither algorithm is significantly better than the other, when examining the AUC scores over all the provided datasets.

This result seems to contradict the paper, which claims that RotBoost perform better. There are several explanations that can be provided in order to understand that:

1. Statistical hypotheses test – in the paper they performed a one-tailed paried t-test with significance level **per dataset**, whereas, we conducted a statistical hypothesis test using Mann-Whitney rank test (two-side-tail test), and setting a significance level **over all datasets**.
2. Hyper-parameters settings - in the paper a classification tree was adopted as the base learning algorithm and the parameters included in the algorithm were all set to be the default values, whereas we use Bayesian hyper-parameters tuning procedure which **span more space** of hyper-parameters, including of the inner base learner.
3. Different experimental settings – in the paper they followed a similar methodology to [Webb, G.I., 2000. MultiBoosting], namely, utilized 20 runs (instead of 10) of threefold cross-validation method to estimate the prediction errors. Then the error rate averaged over the 20 trials to evaluate the performance of that classification method. Whereas, we used nested-cross-validation as described above.
4. Different metric – they based their results upon accuracy metric whereas we used AUC.

Remark: different datasets - the dataset collection that we have, contain datasets that don’t appear in the paper. In order to eliminate this difference, we have conducted another experiment that includes only datasets that appear in the paper and got similar result - no significant difference between the two algorithms performance (p-value=0.93).

In the meta-learning experiment, we found that we were not able to classify which algorithm (RotBoost or Rotation Forest) is better given a specific dataset.

There are several explanations that can be provided in order to understand that:

1. The algorithms performed very similarly across all the datasets.
2. Thorough examination of the AUC scores for the datasets for the two algorithms make it seems like there is no one algorithm which is better on some types of datasets than the other.
3. The fact that there is no one significantly better algorithm makes it not possible to even yield a simple “always predict the majority class” classifier.

Remark: we comment that the procedure by which we assigned the labels (1 in case RotBoost has better AUC average score for a specific dataset and 0 for the opposite) for this task is problematic and prone to noise. It will be interesting to investigate further by adopting a labeling procedure in which we assign 1 in case RotBoost is significantly better for a given dataset, 2 in case Rotation-Forest is significantly better and 3 if there is not significant difference. The learning of this task might be more plausible.

We found that there is no significant difference between the two algorithms (although, Rotation-Forest is seen to outperform RotBoost on more datasets than the reverse).

However, in contrast to Rotation-Forest, its training time is much shorter, probably due to less PCAs computations (in RotBoost there is up to 10 estimators – which means up to 10 PCAs, while in Rotation-Forest there is up to 100 estimators – which means up to 100 PCAs).

Thus, the RotBoost algorithm has a much shorter training time than Rotation Forest, with equivalent performance, which makes it more recommended for use for cases that when training time is important.

**12. Citations**

* [Hyperspectral image remote sensing classification using RotBoost](https://iopscience.iop.org/article/10.1088/1742-6596/1469/1/012095/pdf) -

“In machine learning, the classification of hyperspectral data has several challenges, including high number of dimensions, number of output classes, and limited data references. The solution given to overcome the challenge is to use Ensemble Learning. The benefit of using Ensemble Learning is that we can improve the classification performance of hyperspectral data. One of the Ensemble Learning methods is RotBoost, which is a combination of the Rotation Forest and Adaboost methods. **Experimental results showed that RotBoost produces better accuracy than the Rotation Forest. S and T parameter values are also not very influential on RotBoost accuracy**”.

* [Using Machine Learning for Land Suitability Classification](https://www.ajol.info/index.php/wajae/article/view/121896) -

“In this paper, RotBoost algorithm was applied to tackle the land suitability classification problem and this ensemble classifier generation method is a combination of Rotation Forest and AdaBoost. Results demonstrate that RotBoost algorithm can generate ensemble classifiers with significant higher prediction accuracy than either Rotation Forest or AdaBoost, which is about 99% and 88.5% using two different performance evaluation measures. So, for determination of land suitability class can use RotBoost algorithm instead of other methods. The RotBoost algorithm has more accuracy and faster than the other method for determination of landform classification. To achieve the high classification accuracy by using RotBoost algorithm, this method can be suggested as a robust one for land suitability classification.”

* [A ROBUST MODEL FOR GENE ANLYSIS AND CLASSIFICATION](https://d1wqtxts1xzle7.cloudfront.net/38398770/3111ijma02.pdf?1438854670=&response-content-disposition=inline%3B+filename%3DA_ROBUST_MODEL_FOR_GENE_ANLYSIS_AND_CLAS.pdf&Expires=1597960879&Signature=ceAQ5XBHTPRxsRJZBBux6uYVnKg4K6~D9WNZIsWegkM6kMG-mM8i6kW7DfbuLalAl64WQuninH7ykicgUUCCo4WFY6zd0DngGkcZIcxnWNqhSQpd7epMj0HtDKEV7fsQdvY-WPQZfdJJpbsSnlLOcEb9su7cNUJD8IM0Pa~CVUtMTLYf8ZTmo9~Mq0ZlVVUPhSGaXJvvYxg8vItJ9ZCFH0N0qwASFHTqxG0sNyXWwrpDO29Y24EN3-UMWaSTWVtEk9rtbwbBsgVRt~MGyDGV7NGCUPoDGxPEhCpV9gwmhD-2jDI9yiC9d990V3y7fHBUtJ~Udx1qwTJdCsfeIoCGFA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA) -

“The experimental results show that Rotboost ensemble with several basis classifiers is a robust method for microarray classification, which achieved the highest accuracy for majority of the analyzed benchmark datasets. In fact, RotBoost is found to perform much better than the other examined counterparts. By the way, the improvement of generalization ability achieved by RotBoost is obtained with negligible increase in computational costs. Indeed, RotBoost provides a potential computational benefit over AdaBoost in that it can be executed in a parallel manner”.

* [An empirical evaluation of rotation-based ensemble classifiers for customer churn prediction](https://www.sciencedirect.com/science/article/abs/pii/S0957417411005239) - “In this paper, two rotation-based ensemble classifiers are proposed as modeling techniques for customer churn prediction.  In terms of accuracy, RotBoost outperforms Rotation Forest, but none of the considered variations offers a clear advantage over the benchmark algorithms. However, in terms of AUC and top-decile lift, results clearly demonstrate the competitive performance of Rotation Forests compared to the benchmark algorithms. Moreover, ICA-based Rotation Forests outperform all other considered classifiers and are therefore recommended as a well-suited alternative classification technique for the prediction of customer churn”.

# [Short Term Earthquake Prediction in Hindukush Region Using Tree Based Ensemble Learning](https://ieeexplore.ieee.org/abstract/document/7866782) - “The short term earthquake prediction is performed using tree based ensemble classifiers, where rotation forest has shown good prediction results, compared to random forest and rotboost”