CPTS 437
Investigating the Random Forest Algorithm

Christian Estlund

Vishal Malireddi

Aidan Bachart

Alex Langland

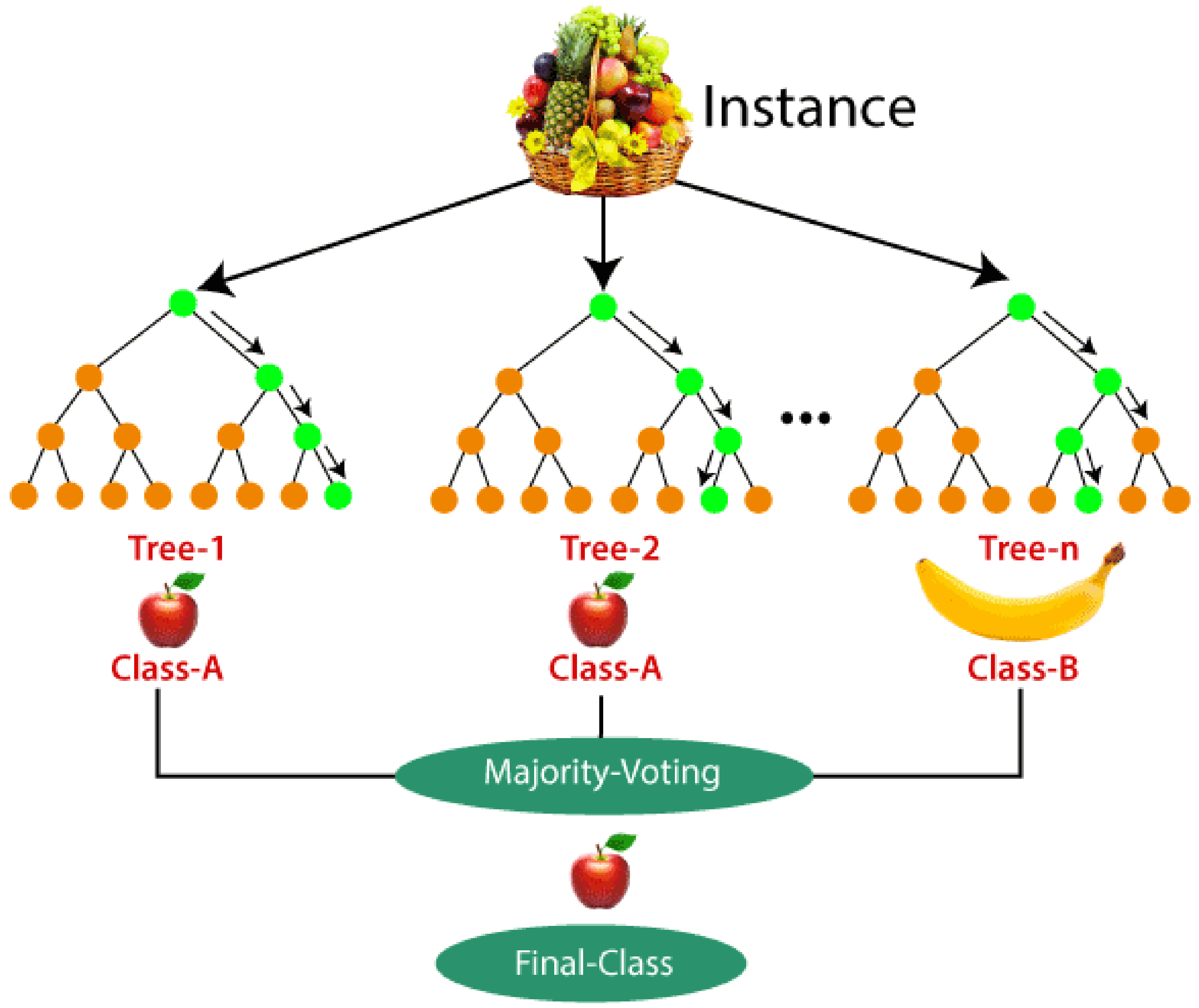
Our Question

We are seeking to understand the uses, efficacy, and workings of the **Random Forest Algorithm**

To investigate this, we built our own implementation of the algorithm and trained it on two data sets to analyze its performance.

The Random Forest Algorithm

- Works by creating an array of decision trees and taking the most common result in an ensemble-voting method
- Different trees are usually built using subsets of the data and features to capture more individual relationships



Why use a Random Forest?

The Random Forest model can be applied where a decision tree can, and comes with some distinct advantages as compared to decision trees:

- More effective when data is high-dimensional
- Far less prone to overfitting

Its main disadvantage is its higher computational complexity and training time.

Our Data

We used **two** datasets to train and test our implementation:

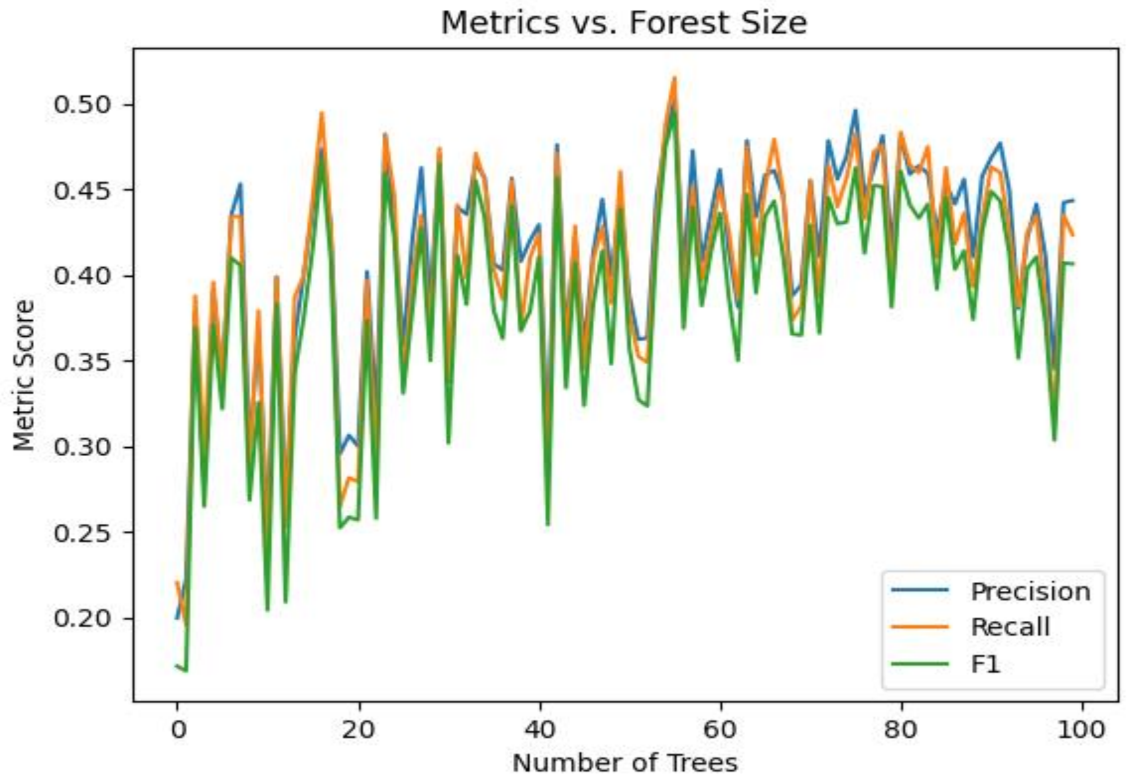
1. A synthetic dataset from Rabie El Kharoua. This contains data entries for over 2,000 high school students, with our target class being the students' academic performance in terms of grade class (0-4). This dataset is very clean and allows us to test our algorithms in favorable conditions.
2. A real dataset from a Portuguese bank. This contains data entries for bank customers, with our target class being whether the customers are subscribed to a term deposit (0 or 1). This dataset is more representative of real-world conditions and gives us more rigor to test our model.

Implementation

First, we preprocess our data to normalize continuous features for ease of training the model.

Then, we train *n* decision trees, each tree being built from a randomly bootstrapped sample of the data and split by a randomly selected subset of features.

Results



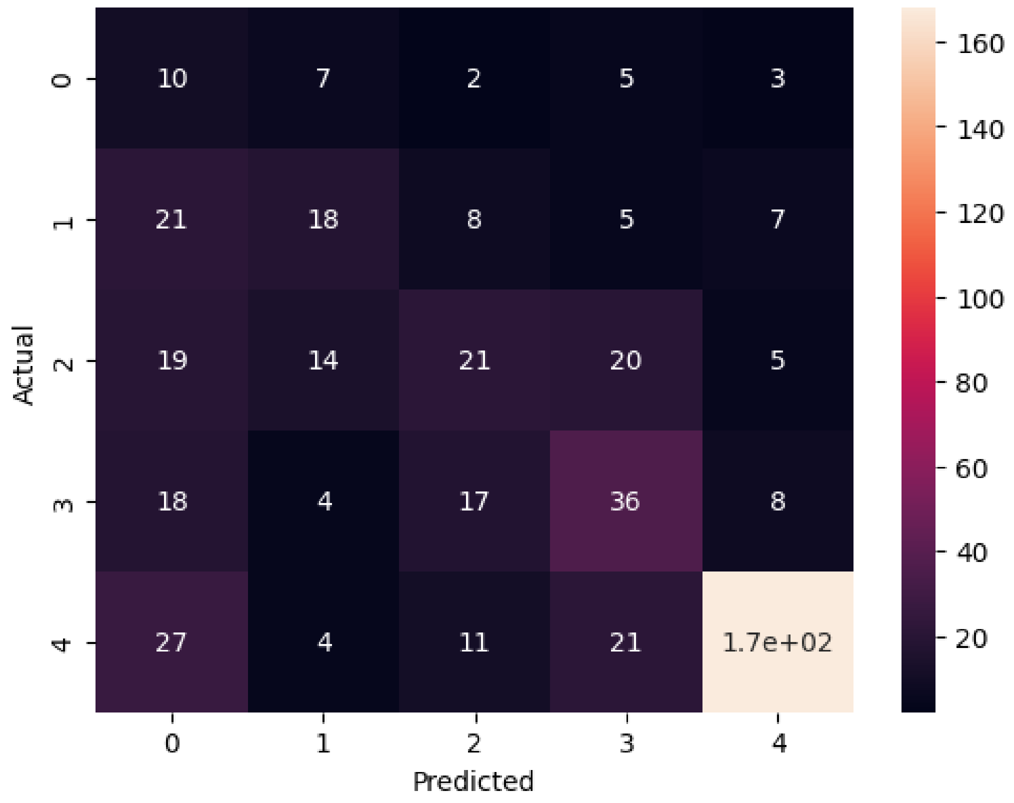
Our algorithm run on the student performance dataset

We tested the performance of the model while increasing the number of trees to be built in the forest.

Although the data is noisy, there is an average increase in all metrics as the number of trees increases.

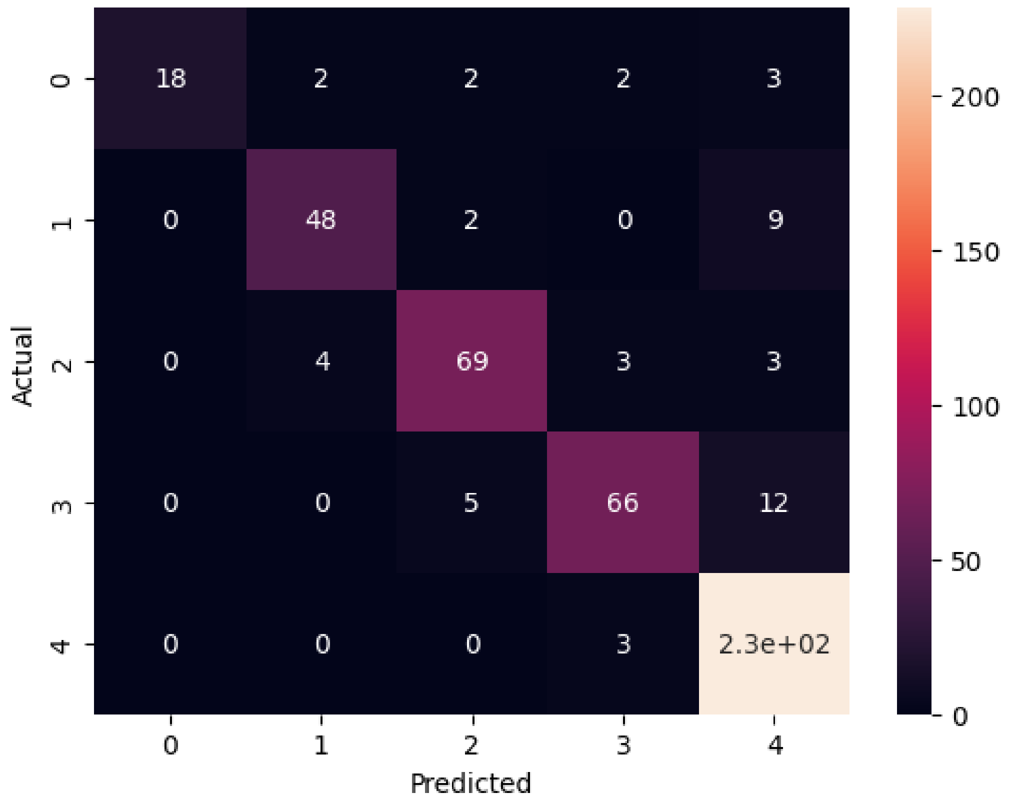
As a test, we also trained our model on the bank's term loan dataset. With 5 trees, this yielded a precision of **0.645**, recall of **0.591**, and an f1 of **0.557**.

Results: Comparative



Actual \ Predicted	0	1	2	3	4
0	10	7	2	5	3
1	21	18	8	5	7
2	19	14	21	20	5
3	18	4	17	36	8
4	27	4	11	21	1.7e+02

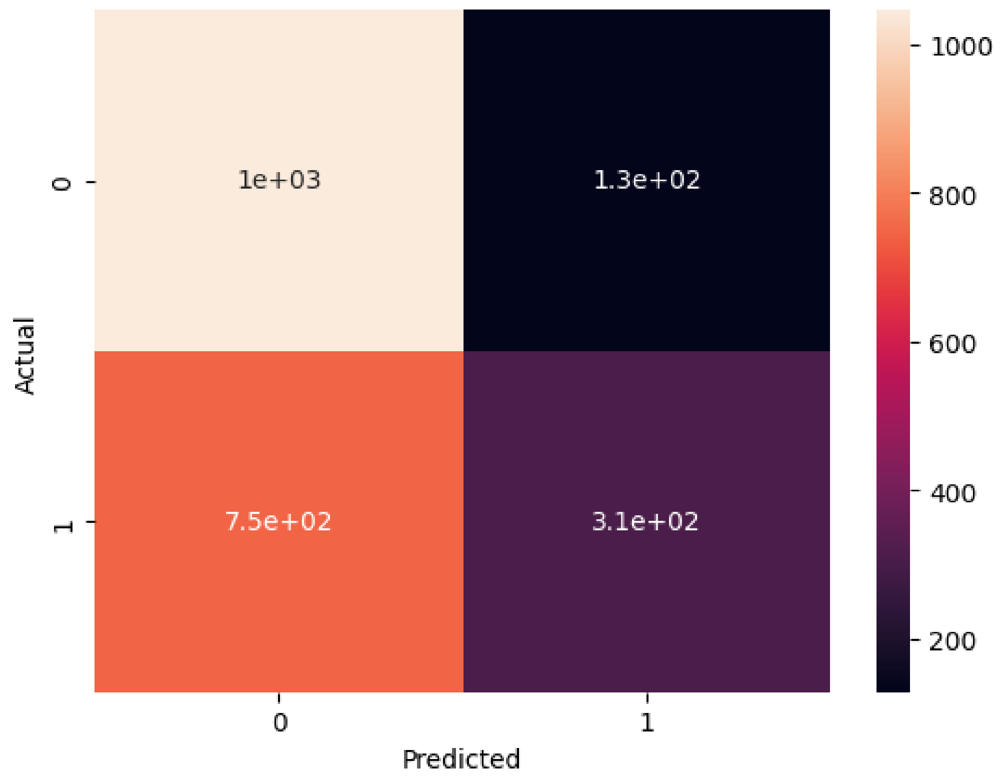
Our algorithm run on the student performance dataset



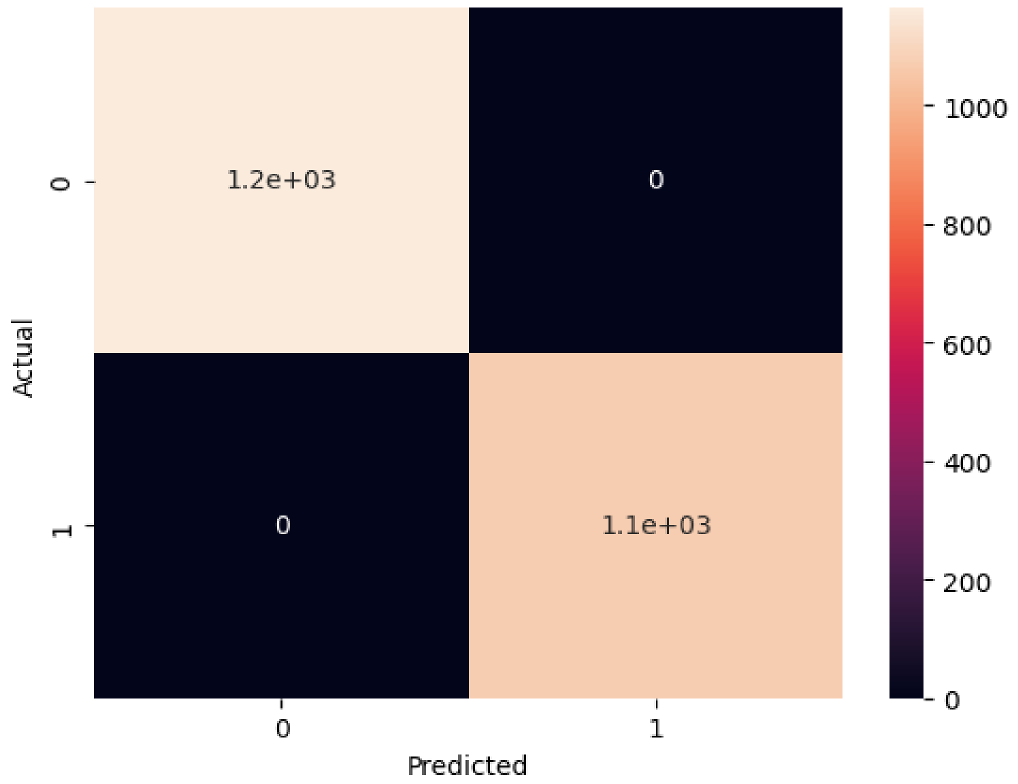
Actual \ Predicted	0	1	2	3	4
0	18	2	2	2	3
1	0	48	2	0	9
2	0	4	69	3	3
3	0	0	5	66	12
4	0	0	0	3	2.3e+02

sklearn's Random Forest run on the student performance dataset

Results: Comparative



Our algorithm run on the term loan dataset



sklearn's Random Forest run on the term loan dataset

Results: Comparative

Across both datasets we tested, our model functioned but was beat out by sklearn's implementation of the same algorithm.

This could be due to various reasons, such as:

- Our simplified handling of continuous variables
- How we determine where to split when building the trees

Conclusion

Although our implementation of the random forest algorithm could be improved, it is clear through our research and the impressive performance of sklearn's implementation that the random forest algorithm is an effective option to make predictions on these datasets.

Thank you!