Decision Trees and Random Forests for Photoluminescence Prediction

CHEMICAL ENGINEERING
UNIVERSITY of WASHINGTON

Wesley Beckner, Matt Murbach, and Janet Matsen

Background

Machine learning and high performance computing revolutionize the strategies for material discovery by leveraging the researcher's attention. By the creation of automated and adaptive experimental architectures, new design spaces can be explored that would otherwise be insurmountable.[1] In this spirit we sought to illustrate the performance of random forests as implemented in sklearn.ensemble.RandomForestRegressor[2] for the prediction of photoluminescence of perovskite solar cells. The tutorial is meant to be a creative exploration space to introduce other researchers and students to one of the most commonly used machine learning algorithms.

Methods

Data for each decision tree in each random forest was pre-computed in Python using scikit-learn. Models were trained on 75% of the data, reserving 25% for testing.

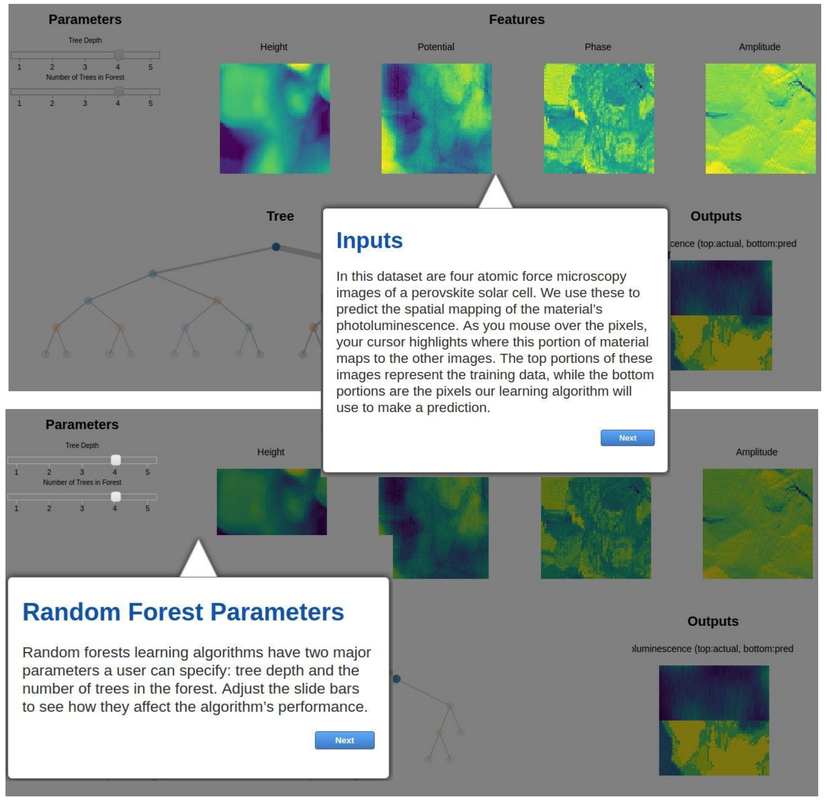
Generated models varied by two hyperparameters: tree depth, and number of trees in each forest. Both parameters were varied in combinations from 1 to 5. Additional depth and trees in each forest did not reduce the mean squared error of the predictions.

The script tree_to_dict.py was developed to convert the tree structure from the scikit-learn class _tree to D3-ready JSON files. This script recurses over the tree structure from the root of a scikit-learn tree, storing decision information and flow through the node in dictionaries. By nesting each child dictionary inside its parent, a tree-like structure was created. These trees were then exported to JSON text files, which were loaded into D3 using the d3.json built-in code.

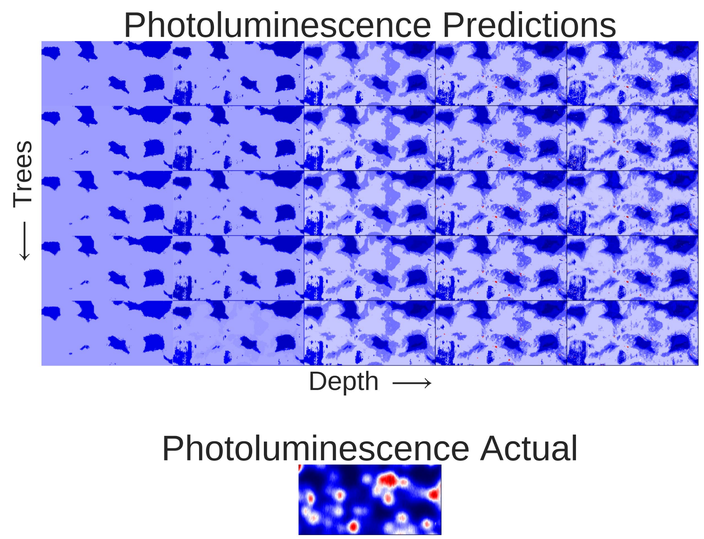
Representing the Ensemble of Forests and Images in D3

Our tutorial depicts two types of datasets that lend themselves well to visual encoding: decision trees and AFM images. Our task was to determine how to direct focus+context to not overwhelm the user.

Tutorial Design



To this end, for the hyperparameters we appropriated slider bars; for the images we used 'viridis' color mapping to differentiate high and low pixel values and tooltips for mapping features together; and for the decision trees colorized nodes indicating feature type being bisected, edge thickness to indicate 'feature traffic' through the branch, and tooltips to see the details in the node decision.



References

[1] Beck, D. A. C., Carothers, J. M., Subramanian, V. R. and Pfaendtner, J. (2016), Data science: Accelerating innovation and discovery in chemical engineering. AIChE J., 62: 1402–1416. doi:10.1002/aic.15192

[2] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

Seattle, WA, June 2016