CLEMSON UNIVERSITY

RANDOM FORESTS
for classification and regression

Presented by Carl Ehrett

2019-04-22

The appeal of random forests

Random forests are popular for their **ease of use** and **minimal tuning required**. They can be applied with default settings and provide **good out-of-the-box results**. They can be used for **nonparametric regression or classification**.

Claims made on Breiman and Cutler's website[1] include:

- It is **unexcelled in accuracy** among current algorithms.
- It gives estimates of what variables are important in the classification.
- It generates an internal **unbiased estimate of the generalization error** as the forest building progresses.
- **Random forests does not overfit.**

Some of these claims are even true.

1 https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#overview

What are random forests?

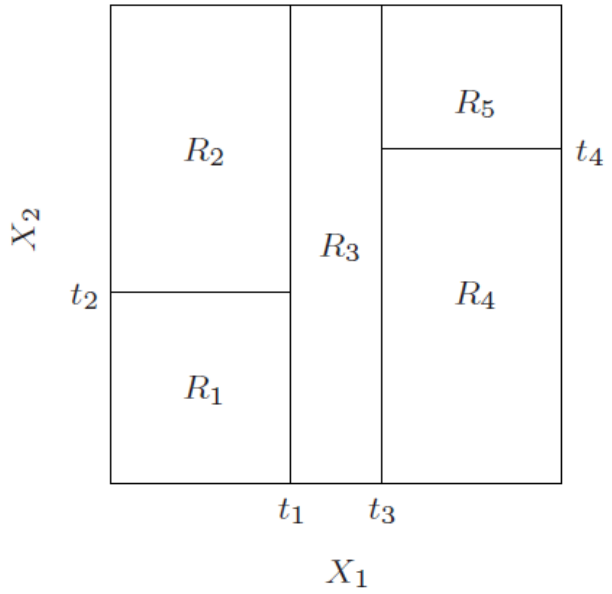
Random forests are an **ensemble learning** method which uses **bagging** in conjunction with **random feature selection** to reduce the variance of a **tree** estimator.

We will discuss each of the underlined terms above.

Ensemble learning and trees

An ensemble learning method is one in which a collection of *weak learners* are combined, often through averaging (for regression) or voting (for classification).

In the case of random forests, the weak learners are *decision trees*. The trees used in random forests can be grown in many ways, often following the CART method.



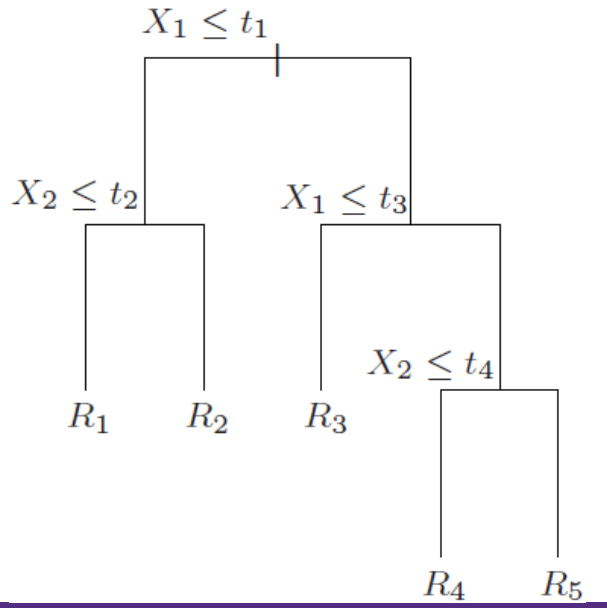


Image credit: Hastie, T., Tibshirani, R., and Friedman, J. (2009). The Elements of Statistical Learning. Springer Series in Statistics, p. 306.

Bagging

Decision trees are prone to having high variance. To mitigate this problem, we can use bootstrap aggregation, a.k.a. *bagging*.

Suppose you have a method for producing an estimator $\hat{f}(x)$ at a point x using your training data. To produce a *bagged* version of this estimator, you would:

- Produce B bootstrap samples of your training data,
- Fit your model on each bootstrap sample,
- Get a prediction $\hat{f}^{*b}(x)$ using each bootstrap sample fit $b = 1, \ldots, B$, and
- Combine all of these predictions.

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x) \text{ or } \hat{f}_{\text{bag}}(x) = \text{argmax}_k \sum_{b=1}^{B} \mathbf{1}_{\hat{f}^{*b}(x)=k}$$

Random feature selection

Bagging reduces variance, but this reduction is limited by the level of correlation amongst the bagged predictors. The variance of the bagged estimate is[1]:

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

As $B \to \infty$, the second term goes to 0, but the first term remains constant. So bagging can't help us reduce this first term.

To reduce the first term, we use random feature selection. When growing the decision trees, every time a node is split, a random subset of the predictors are chosen to be candidates for splitting.

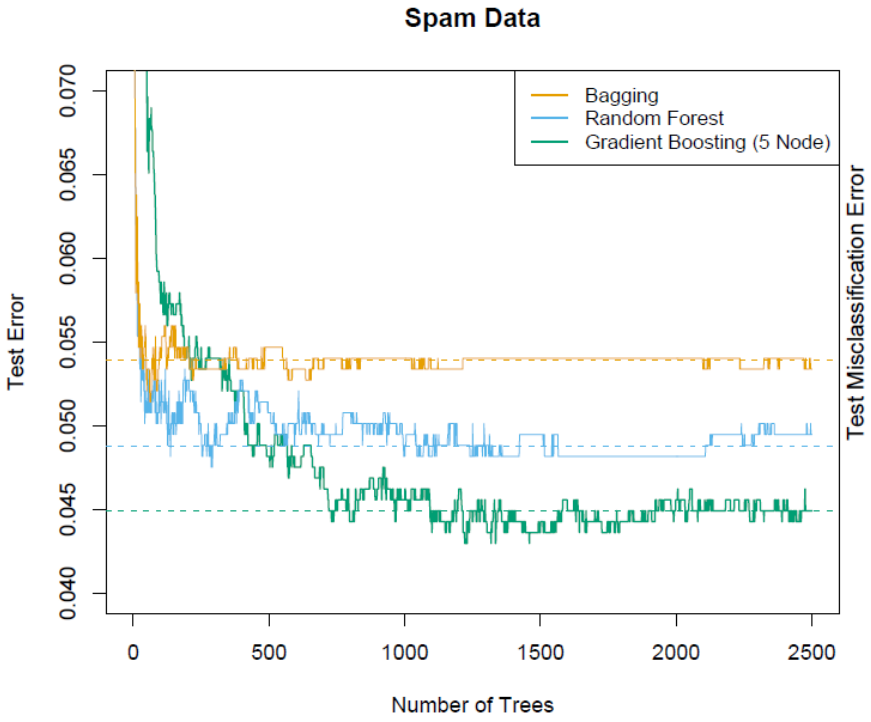
The result is to reduce correlation amongst the trees, and thus to reduce ρ.

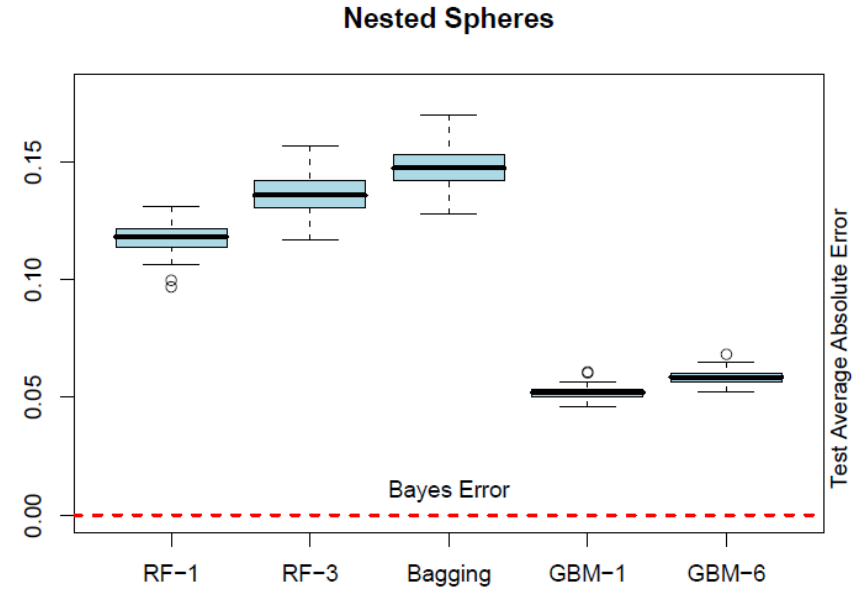
1 Hastie, T., Tibshirani, R., and Friedman, J. (2009). The Elements of Statistical Learning. Springer Series in Statistics, p. 588.

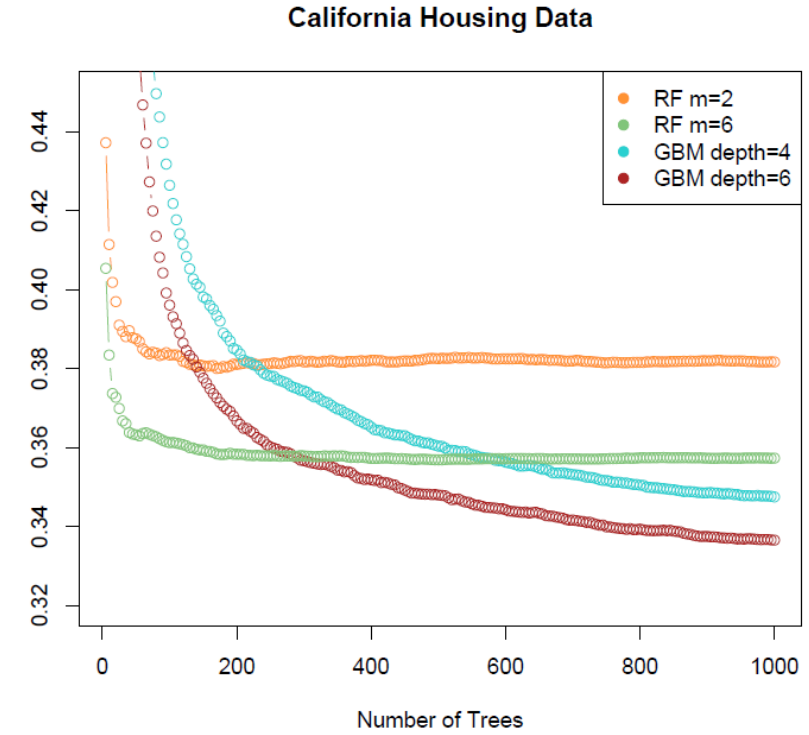
Claims about random forests: unexcelled?

Breiman and Cutler: “It is unexcelled in accuracy among current algorithms.”

Hastie, Tibshirani and Friedman[1]:







However, see also Fernández-Delgado, M., Cernadas, E., and Barro, S. (2014). *Do we need hundreds of classifiers to solve real world classification problems?* Journal of Machine Learning Research. 15, pp 3133-3181, where a pretty strong case is made for random forests being top of the heap.

1 (2009). The Elements of Statistical Learning. Springer Series in Statistics, pp. 589, 590, and 591, respectively.

No test sets, no cross validation, no problem

A big selling point of random forests is that it provides an estimate of the generalization error (i.e. test error) without requiring a test set or even cross-validation.

Instead, the **OOB error** (out-of-bag error) is used to estimate generalization error.

To get the OOB error: For each training sample, test it on the trees *that don't include that sample in their bagged subset of the training data*.

Feature importance

Random forests make it easy to estimate the relative importances of your covariates. This can be done in multiple ways.

- For each node split, record the improvement (e.g. measured via Gini impurity) generated by the split. For each covariate, average the improvements it produces each time it is used for a split[1].
- For each tree, find the OOB error, then for each covariate measure the increase in OOB error when you randomly permute that covariate's values. Average these values across all trees[2].
- For each node split, record the proportion of training samples that are "downstream" of that split. For each variable, find the average proportion of samples that are "downstream" of its splits[3].

1 Hastie, T. et al. (2009). 2 Breiman, L. (2001). 3 Pedregosa, F. et al. (2011).

Random forests does not overfit?

Any method can overfit.

When it is claimed that random forests do not overfit (or, more carefully, that they are resistant to overfitting), the reality behind that statement is usually that *adding more trees* does not cause random forests to overfit.

This resistance to overfitting doesn't mean you don't need to be careful about which predictors you include in your model.

Application: facial recognition

- Goal: construct a facial recognition system that uses a low number of covariates.
- For this, I used the Python OpenCV library to capture and manipulate webcam input, and the Python Dlib library for facial landmark detection.
- The landmarks are the 2-D coordinates of the 68 points shown in this image.

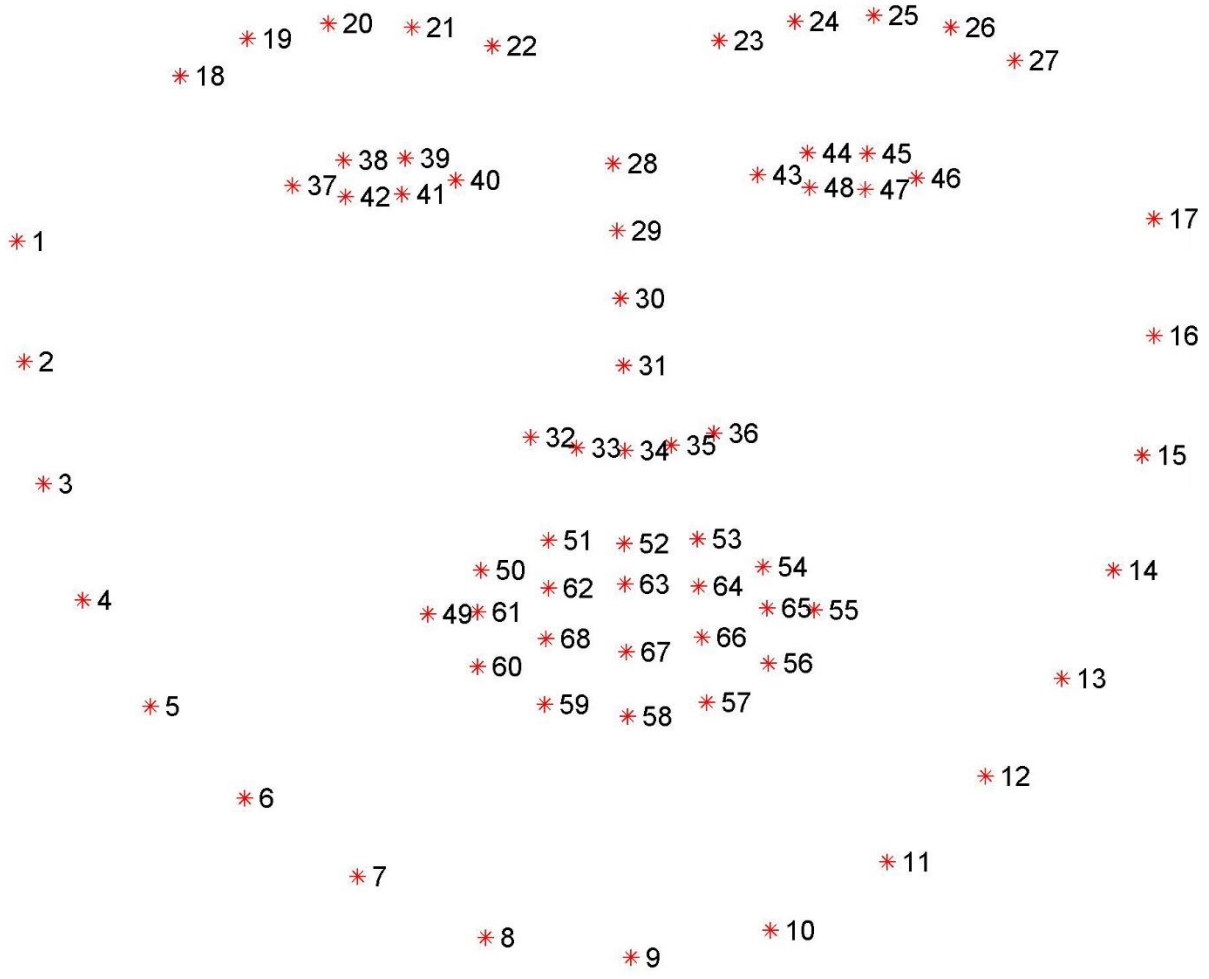
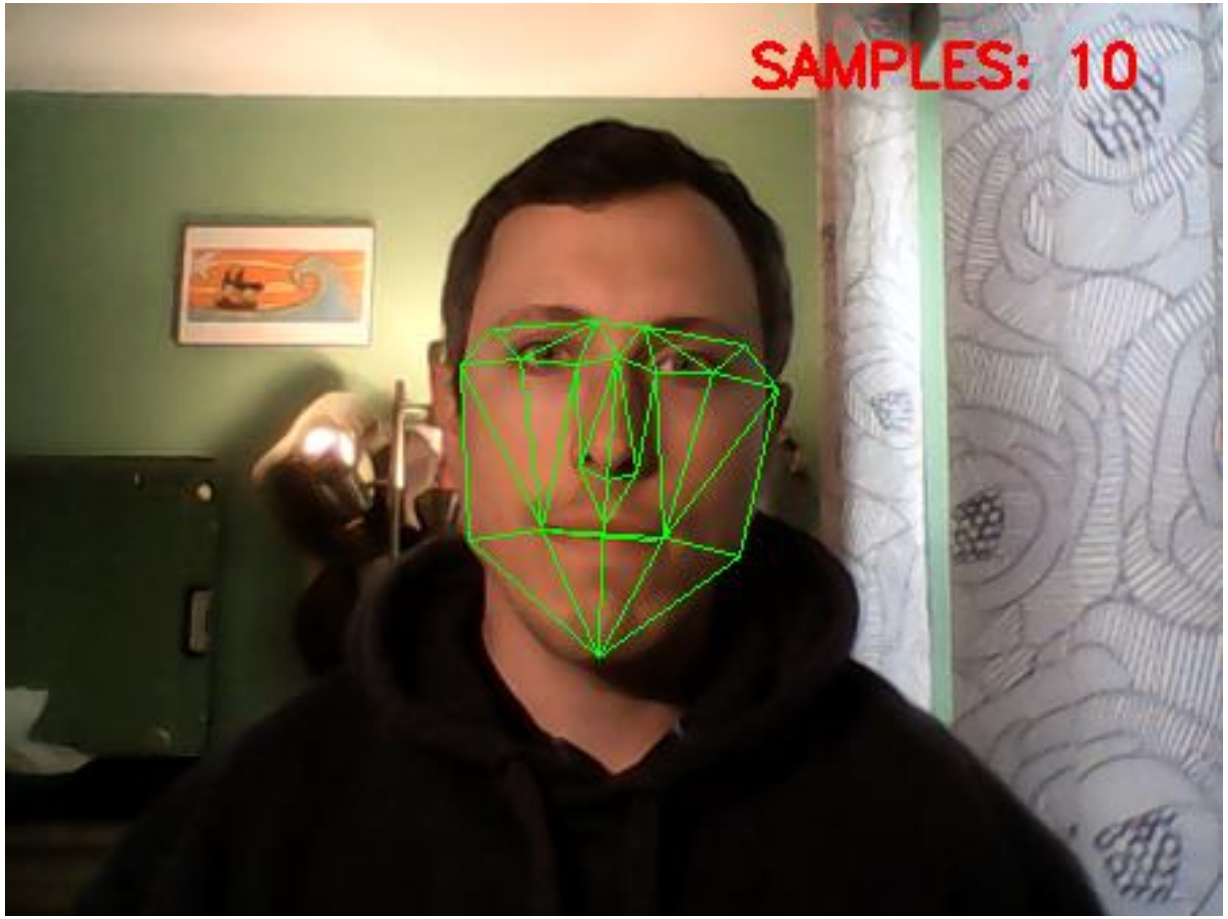


Image credit: Intelligent Behaviour Understanding Group (iBUG), Department of Computing, Imperial College London, https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/

Covariates

I used the 68 landmarks to define 30 covariates. Each covariate is either a length (between two landmarks), the area of a triangle formed by three landmarks, or the angle formed by three landmarks.

To train the random forests model, I gathered 500 samples each from myself and from my wife, using my laptop's webcam.



Fitting the model

- The algorithm performed well with default settings of 10 trees (~6% OOB error).
- However, I found that better performance was possible up to about 100 trees (~3% OOB error), after which there were rapidly diminishing returns.
- Changing number of features, maximum tree depth, etc. had little effect.

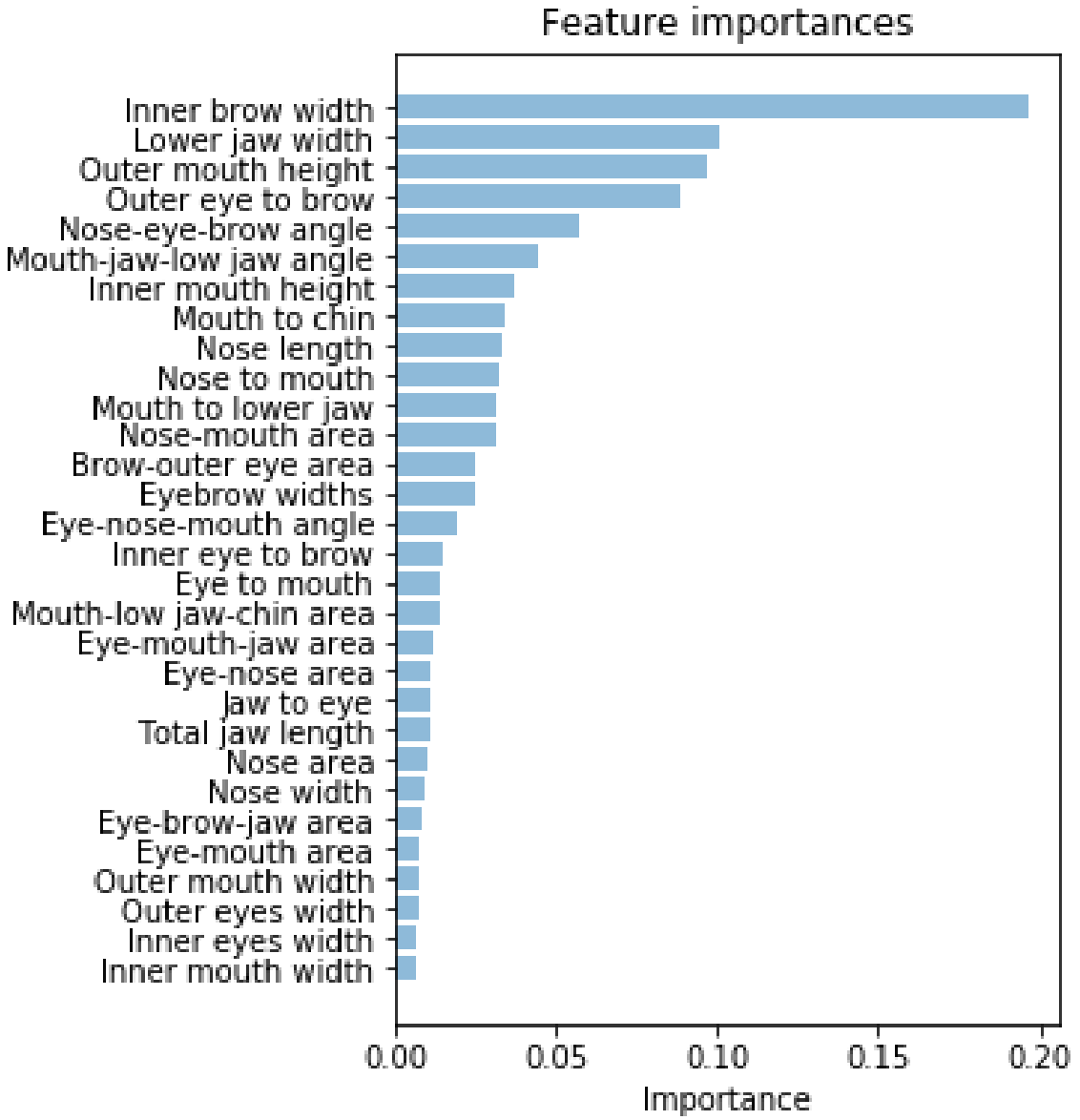
```
>>> from sklearn.ensemble import RandomForestClassifier
>>> import numpy as np
>>>
>>> # Load data
... X = np.zeros([1000,30])
>>> X[0:500,] = np.load(('C:\\Users\\carle\\Documents\\'
...  'Python\\Math 9810 Machine Learning\\carl_train.npy'))
>>> X[500:1000,] = np.load(('C:\\Users\\carle\\Documents\\'
...  'Python\\Math 9810 Machine Learning\\erin_train.npy'))
>>> # Make response vector
... Y = np.zeros([1000,1])
>>> Y[500:1000] = 1
>>>
>>> # Make classifier
... clf = RandomForestClassifier(n_estimators=100,oob_score=True)
>>>
>>> # Produce model
... clf.fit(X,Y.ravel())
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=True, random_state=None, verbose=0, warm_start=False)
>>>
>>> # Check OOB score
... clf.oob_score_
0.974
```

Results

- The classifier performs reasonably well.
- Misclassifications occur with frequency similar to that suggested by the OOB error.
- The classifier is lightweight enough to function in real time on multiple subjects.

Feature importances

- Somewhat unexpectedly, by far the most important feature for distinguishing my face from my wife's appears to be the distance between our eyebrows.
- Lower jaw width is, less surprisingly, also important.
- Outer mouth height's high position on the list is likely an artifact of the data collection process.



Secondary application

Since the classifier is trained only on two individuals – myself and my wife – it serves as a *de facto* classifier for whether someone more closely resembles me or her.

However, the results of attempting to use it for this purpose were inconclusive.