Random Forest

Edgar Acuña

Department of Mathematical Sciences

UPR-Mayagüez

April 2019

Introduction

- **Random forest** is a combination of many decision trees and whose predicted class is obtained by choosing the mode of the classes predicted by individual trees
- Introduced by L. Breiman, 2001.
- This method combines “bagging” with the random selection of predictive atributes.

Algorithm-1

1. Consider a training set with N rows and M predictive variables
2. Build n training samples (usually 500) for the tree choosing samples with replacement and the same size as the original sample (bootstrap samples). The instances that are not chosen form a test sample and are used to estimate the error rate of the tree by predicting its classes (out-of-bag estimation of the classification error)
3. Build a decision tree for each bootstrap sample, but in each partition we only use m attributes, chosen at random to secure the best partition.

$$\text{Usually m} = \sqrt{M}$$

Algorithm-2

4. Unlike a trees classifier where we can "prune", here the tree is allowed to grow until the end and no pruning is done.
5. To predict a new instance, it is run through the decision tree and assigned the label of the corresponding terminal node. This procedure is repeated on all the trees and in the end the class predicted by voting is assigned.

Other aspects of RF

- RF considers the fact that the data set has unbalanced classes (Churn problem).
- RF calculates proximities between pairs of data. This is defined as the proportion of the times that two instances fall on the same terminal node in different trees.
- These proximities can be used for clustering, outlier detection, to impute missing values (knn-impute) or also to visualize the data (it is combined with multidimensional scaling)
- It also serves to detect, if there are interactions between predictor variables.

Estimating the importance of each predictor

1. Build S Random Forest. For k=1…S, repeat steps 2 to 4.
2. For each tree that belongs to the kth RF, consider the OOB_t estimate of the misclassification error
3. For each predictor X_j, permute the values of X_j at random to generate a new sample and obtain a new error $OOB_{t_{(j)}}$ using this new sample.
4. An important measure of the predictor X_j is the average difference between $OOB_{t_{(j)}}$ and OOB_t over all trees of the kth RF.
5. An average measure of importance of the predictor X_j will be the average of the measurements of step 4 in all S random Forests.

Random Forest using scikit-learn

```
url= "http://academic.uprm.edu/eacuna/diabetes.dat"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = pd.read_table(url, names=names,header=None)
#La variable de respuesta y debe ser binaria (0,1)
y=data['class']-1
X=data.iloc[:,0:8]
clf = RandomForestClassifier(n_estimators=50,max_depth=10, oob_score=True,random_state=0)
clf.fit(X, y)
print "The accuracy estimated is", clf.score(X,y)
```

```
The accuracy estimated is 0.986979166667
```

```
#Estimating the accuracy by cross-validation
rfmodel=clf.fit(X, y)
scores = cross_val_score(rfmodel, X, y, cv=10)
```

Random Forest using scikit-learn(cont)

```
scores
print("CV Accuracy: %0.3f (+/- %0.3f)" % (scores.mean(), scores.std() * 2))
#Accuracy estimated using  out-of-Bag
print clf.oob_score_
0.7421875
```

Finding the most important features

```
print(clf.feature_importances_)
[ 0.08178093 0.25522871 0.08433383 0.07064764 0.06914087 0.17326173 0.12167218 0.14393411]
```

The most important features are plas mass and age

" "

Random Forest using h2o

```
diabetes = h2o.import_file("https://academic.uprm.edu/eacuna/diabetes.dat")
myx=['C1','C2','C3','C4','C5','C6','C7','C8']
diabetes['C9']=diabetes['C9'].asfactor()
myy="C9"
model=H2ORandomForestEstimator(ntrees=50,nfolds=10,max_depth=10)
model.train(myx, myy, training_frame = diabetes)
y_pred=model.predict(diabetes)
print (y_pred['predict']==diabetes['C9']).mean()
[0.9752604166666666]
model.model_performance(diabetes)
```

Random Forest using h2o(cont)

```
ModelMetricsBinomial: drf
** Reported on test data. **
MSE: 0.0428155400192
RMSE: 0.206919163006
LogLoss: 0.18842735731
Mean Per-Class Error: 0.0117313432836
AUC: 0.999526119403
Gini: 0.999052238806
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.466578477621:
Maximum Metrics: Maximum metrics at their respective thresholds
```

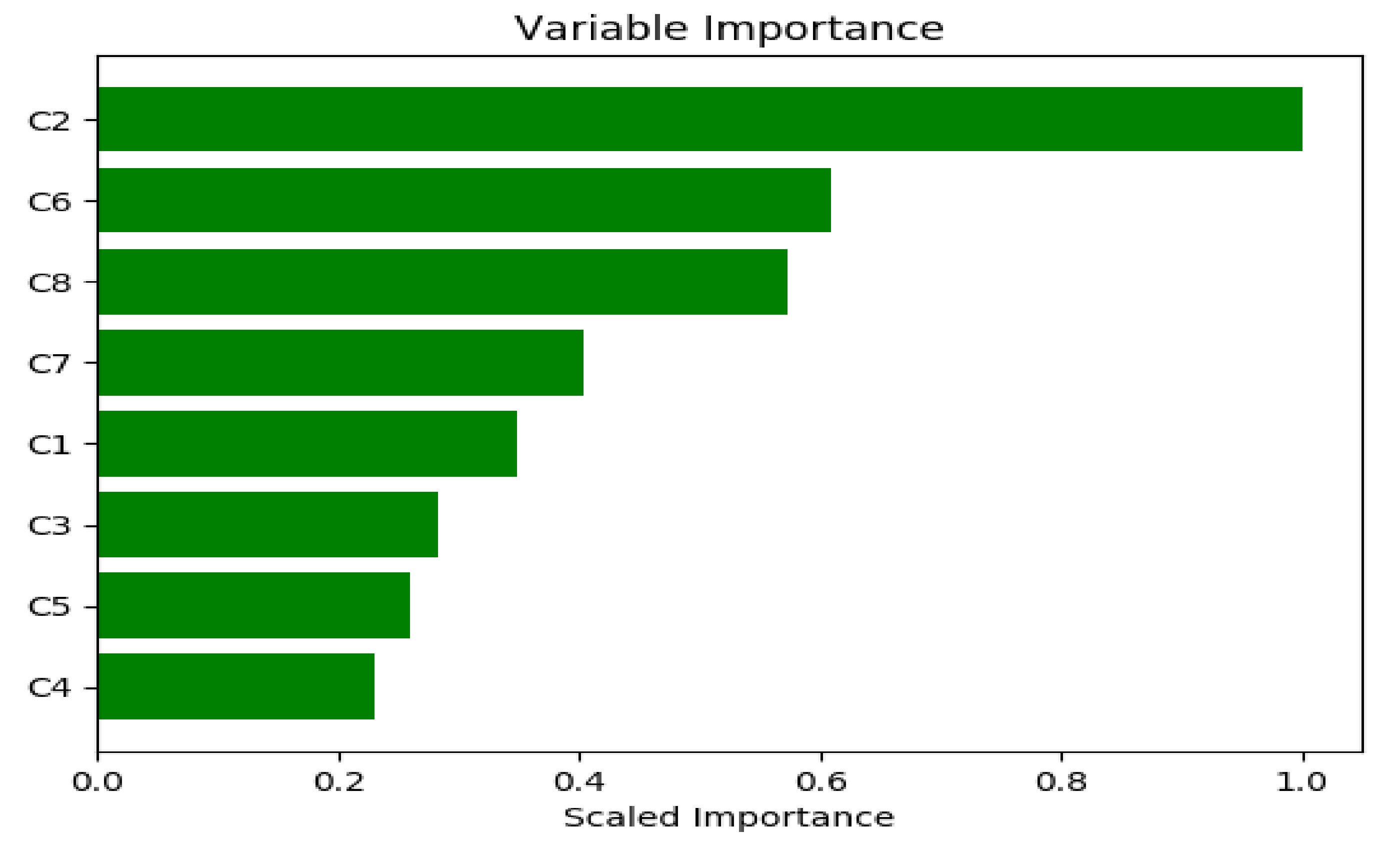
	1	2	Error	Rate
1	497.0	3.0	0.006	(3.0/500.0)
2	6.0	262.0	0.0224	(6.0/268.0)
Total	503.0	265.0	0.0117	(9.0/768.0)

Random Forest using h2o (cont)

Finding the importance of each feature

```
import matplotlib.pyplot as plt
plt.rcdefaults()
fig, ax = plt.subplots()
variables = model._model_json['output']['variable_importances']['variable']
y_pos = np.arange(len(variables))
scaled_importance =model._model_json['output']['variable_importances']['scaled_importance']
ax.barh(y_pos, scaled_importance, align='center', color='green', ecolor='black')
ax.set_yticks(y_pos)
ax.set_yticklabels(variables)
ax.invert_yaxis()
ax.set_xlabel('Scaled Importance')
ax.set_title('Variable Importance')
plt.show()l
```

Random Forest using h2o (cont)



Pros of using RF

- It is one of the Machine Learning algorithms that produces a high level of precision. For many data sets gives the highest level of accuracy.
- Works efficiently with large datasets
- You can manipulate data with thousands of variables, without making previous selection (Bioinformatics). Strong competitor of SVM.
- Estimate the variables that are most important for the classification.
- Calculate an unbiased estimator of the error as more trees are generated.
- It has an efficient method of estimating missing values and maintains accuracy even with a large percentage of missing data (using proximity measurements)

Cons of using RF

- It has been observed that random forests overfits some sets that contain noisy variables.
- For data that includes categorical variables with different number of levels, random forests are biased in favor of those attributes with the highest number of levels. Therefore, the scores given by the RF are not very reliable for this type of data.

Summary and some conclusions:

- RF is very fast
  - RF is quick to build. Even faster to predict!
  - In practice, it does not require cross-validation, it makes you gain speed of training in a factor that can exceed 100
  - It is completely parallelizable, which makes it even faster!
- Automatically includes selection of variables.
- RF is resistant to overfitting.
- RF has the ability to analyze data without the need for pre-processing
  - The data does not need to be rescaled, transformed or modified
  - It is resistant to outliers
  - The treatment of missing values is included in the algorithm.
- It can be used to form clusters using the proximity matrix generated by RF.