Random Forest Modeling

Howard Nguyen

2022-10-02

RANDOM FOREST MODELING

Random Forest is a popular machine learning algorithm that belongs to the family of ensemble learning methods. It is used for both classification and regression tasks and can handle both numerical and categorical data.

The basic idea behind Random Forest is to build multiple decision trees and combine their predictions to get a more accurate and stable model. Each decision tree in the forest is built on a random subset of the training data and a random subset of the input features. This randomness helps to reduce overfitting and improve the generalization performance of the model.

Here are the general steps to build a Random Forest model:

1 - Collect and preprocess the data: Collect the data and preprocess it by cleaning, transforming, and encoding the features. Split the data into training and test sets.

2 - Build decision trees: Build multiple decision trees on different subsets of the training data and features. Each decision tree should be independent and have some randomness.

3 - Make predictions: For each test data point, pass it through each decision tree in the forest to get a prediction. For classification tasks, take the majority vote of all the predictions; for regression tasks, take the average of all the predictions.

4 - Evaluate the model: Evaluate the model on the test set using appropriate metrics such as accuracy, precision, recall, F1 score, or mean squared error, etc.

5 - Tune the hyperparameters: Tune the hyperparameters of the model such as the number of trees, maximum depth, minimum sample split, etc., using cross-validation and grid search to get the best performance.

6 - Deploy the model: Once you have a good model, deploy it in production and use it to make predictions on new data.

Overall, Random Forest is a powerful and versatile algorithm that can be used in a wide range of applications such as healthcare, finance, marketing, and more.

INSTALL AND LOAD PACKAGES

```r
# Install pacman ("package manager") if needed
if (!require("pacman")) install.packages("pacman")
```

```r
# pacman must already be installed; then load contributed
# packages (including pacman) with pacman
pacman::p_load(caret, magrittr, pacman, parallel,
  randomForest, rio, tidyverse)
```

```
# caret: for decision trees
# magrittr: for pipes
# pacman: for loading/unloading packages
# parallel: for parallel processing
# randomForest: for random forests (obviously)
# rio: for importing data
# tidyverse: for so many reasons
```

LOAD AND PREPARE DATA

```
# Set random seed
set.seed(313)
# Import data, select random sample
df <- import("../data/b5_df.rds") %>%
  select(gender, Extrav:Open) %>%
  sample_n(500) %>%                 # Sample 500 cases
  print()
```

```
## # A tibble: 500 x 6
##    gender Extrav Neurot Agree Consc  Open
##    <fct>   <dbl>  <dbl> <dbl> <dbl> <dbl>
##  1 Female    3.7    2.8   4     3.4   3.5
##  2 Female    2.6    4     3.6   4.7   4.1
##  3 Male      3.6    2.3   4.4   3     3.6
##  4 Female    4      3.6   4.1   2.9   3.5
##  5 Male      2.8    2.6   3.6   3.5   4.4
##  6 Male      2.3    3.3   3.3   3.4   3
##  7 Female    2.1    4.2   4.1   4.2   2.9
##  8 Female    3.5    2.4   4.5   4     3.5
##  9 Female    3.3    3.3   3.8   4.4   4.3
## 10 Female    2.5    4     4.3   4.1   4.6
## # ... with 490 more rows
```

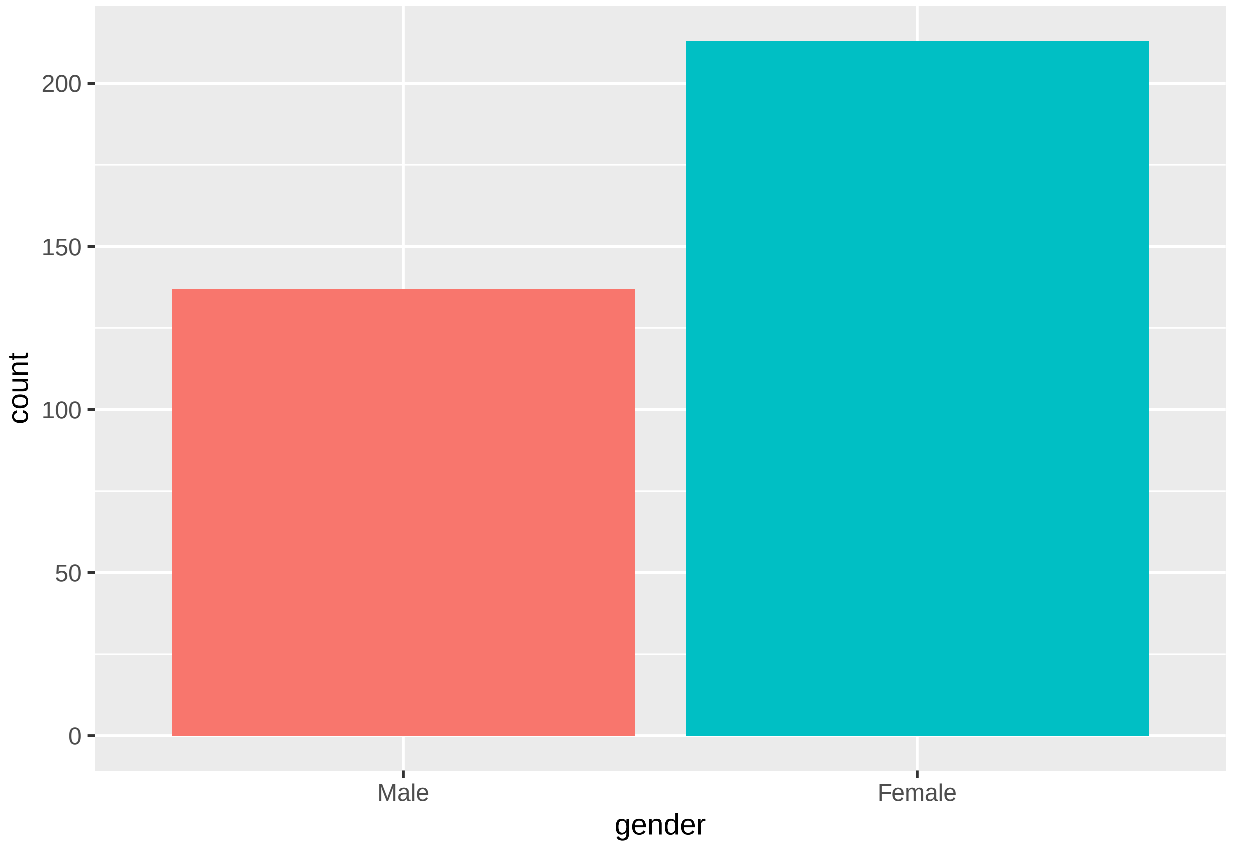
SPLIT DATA

```
# Split data into train and test sets
train <- df %>% sample_frac(.70)
test <- df %>% anti_join(train)
```

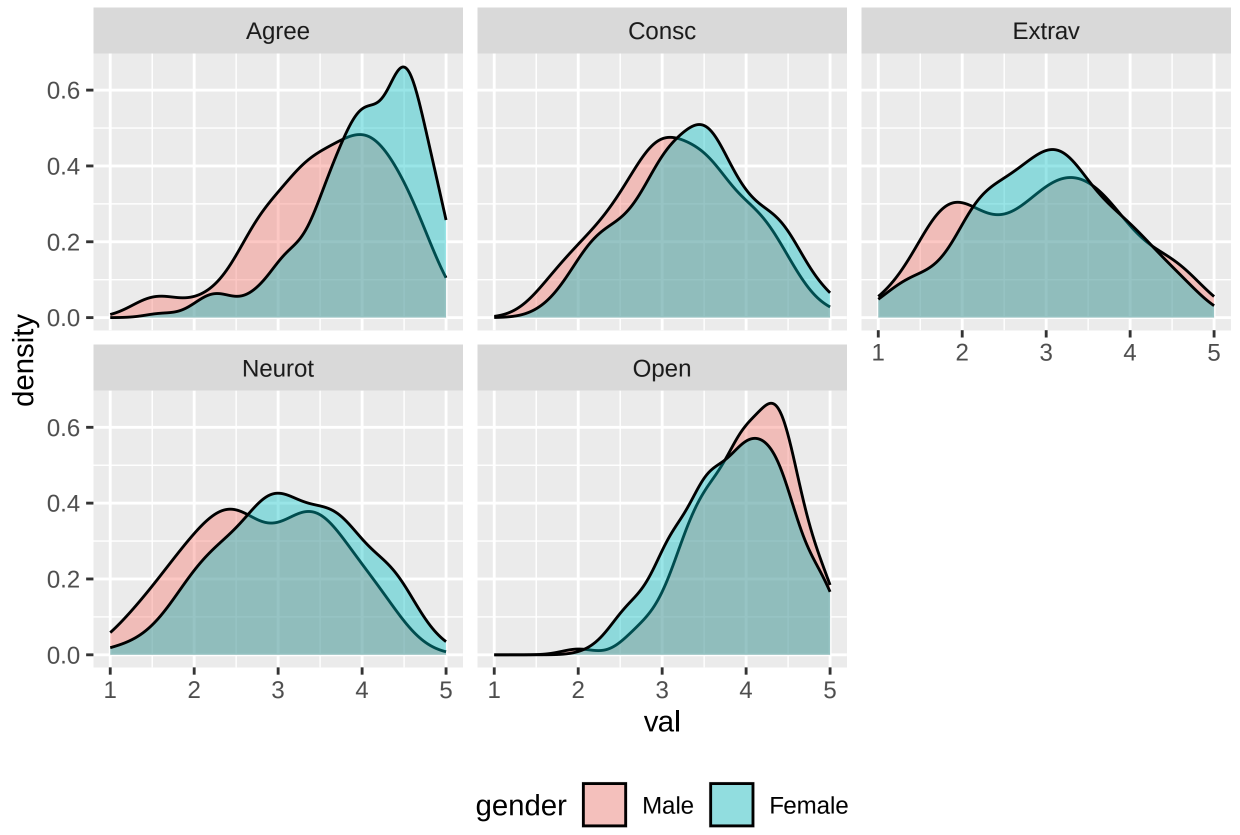
```
## Joining with `by = join_by(gender, Extrav, Neurot, Agree, Consc, Open)`
```

EXPLORE TRAINING DATA

```
# Bar chart of "gender"
train %>%
  ggplot() +
  geom_bar(aes(x = gender, fill = gender)) +
  theme(legend.position = "none")
```



```
# Density plots of Big 5 variables
train %>%
  gather(var, val, Extrav:Open) %>%
  ggplot(aes(val, group = gender, fill = gender)) +
  geom_density(alpha = 0.4) +
  facet_wrap(~var) +
  theme(legend.position = 'bottom')
```



MODEL TRAINING DATA

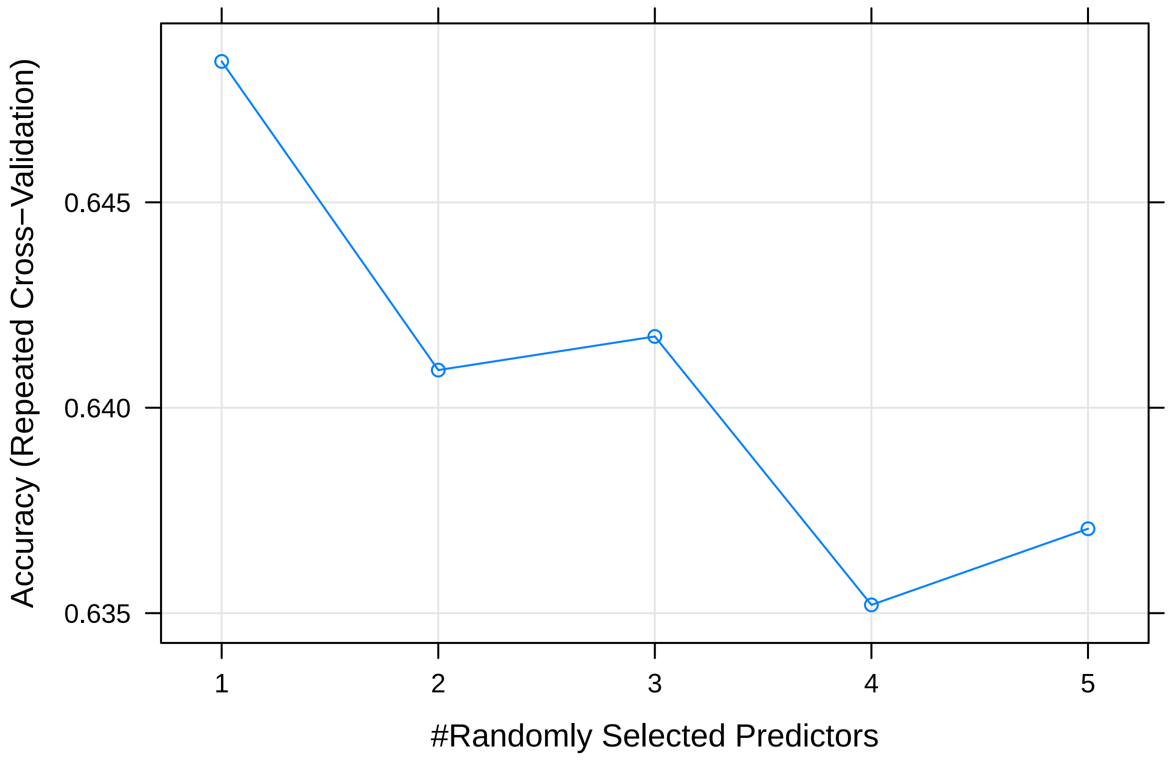
```
# Define parameters for the train function
control <- trainControl(
  method  = "repeatedcv",  # Repeated cross-validation
  number  = 10,            # Number of folds
  repeats = 3,             # Number of sets of folds
  search  = "random",      # Max number of tuning parameters
  allowParallel = TRUE     # Allow parallel processing
)
```

```
# Train decision tree on training data (can take a while)
rf <- train(
  gender ~ . ,           # Predict gender from all other vars
  data = train,          # Use training data
  method = "rf",         # Use random forests
  metric = "Accuracy",   # Use accuracy as criterion
  tuneLength = 15,       # Number of levels for parameters
  ntree = 800,           # Number of trees
  trControl = control    # Link to parameters
)
```

```r
# Show processing summary
rf
```

```
## Random Forest
##
## 350 samples
##   5 predictor
##   2 classes: 'Male', 'Female'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 315, 314, 315, 316, 315, 315, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   1     0.6484298  0.2205045
##   2     0.6409166  0.2155382
##   3     0.6417351  0.2279157
##   4     0.6352007  0.2147962
##   5     0.6370542  0.2192244
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 1.
```

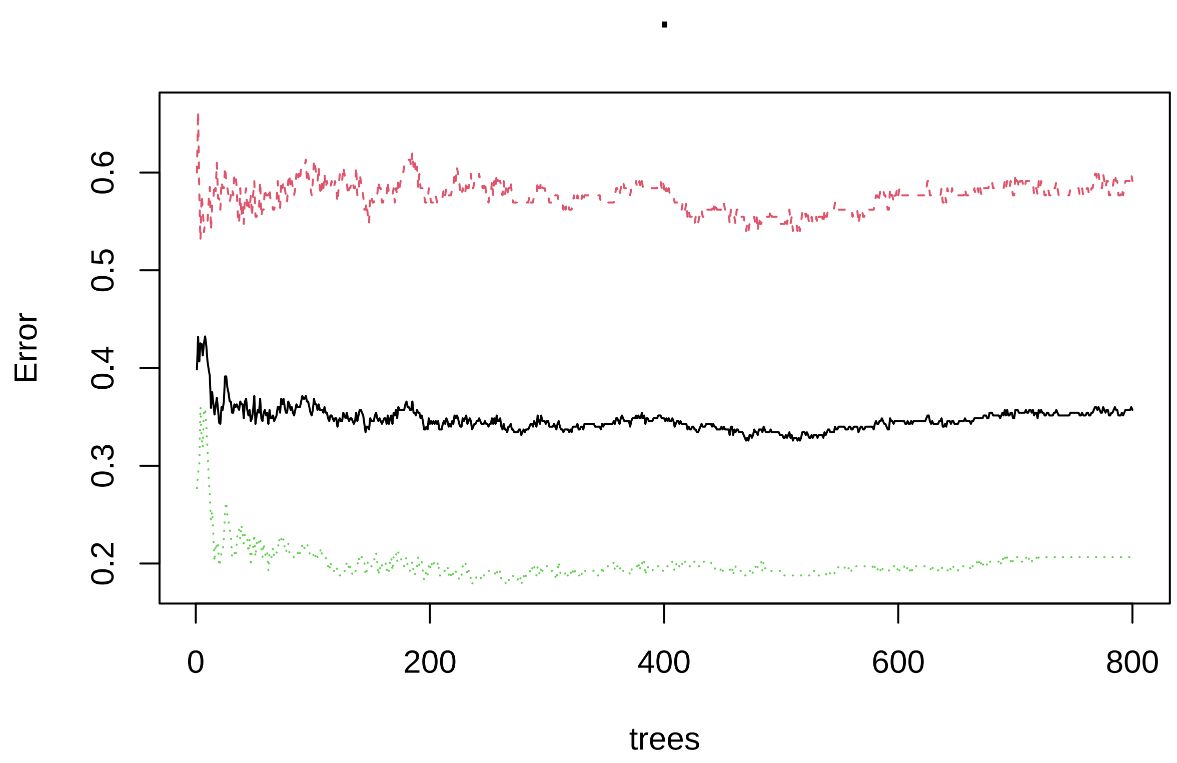
```r
# Plot accuracy by number of predictors
rf %>% plot()
```



```r
# Accuracy of model with training data
rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 800, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 800
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 35.71%
## Confusion matrix:
##        Male Female class.error
## Male     56     81   0.5912409
## Female   44    169   0.2065728
```

```r
# Plot error by number of trees; Red is error for "Male,"
# green is error for "Female," and black is error or "OOB,"
# or "out of bag" (i.e., the probability that any given
# prediction is not correct within the test data, or the
# overall accuracy)
rf$finalModel %>% plot()
```



APPLY MODEL TO TEST DATA

```
# Predict test set
gender_p <- rf %>%          # "predicted"
  predict(newdata = test)   # Use test data
```

```
# Accuracy of model on test data
table(
  actualclass = test$gender,
  predictedclass = gender_p
) %>%
confusionMatrix() %>%
print()
```

```
## Confusion Matrix and Statistics
##
##              predictedclass
## actualclass Male Female
##      Male     22     33
##      Female   24     71
##
##                Accuracy : 0.62
##                  95% CI : (0.5372, 0.6979)
##      No Information Rate : 0.6933
##      P-Value [Acc > NIR] : 0.9776
##
##                    Kappa : 0.1526
##
##   Mcnemar's Test P-Value : 0.2893
##
##              Sensitivity : 0.4783
##              Specificity : 0.6827
##           Pos Pred Value : 0.4000
##           Neg Pred Value : 0.7474
##               Prevalence : 0.3067
##           Detection Rate : 0.1467
##     Detection Prevalence : 0.3667
##        Balanced Accuracy : 0.5805
##
##         'Positive' Class : Male
##
```

The End!