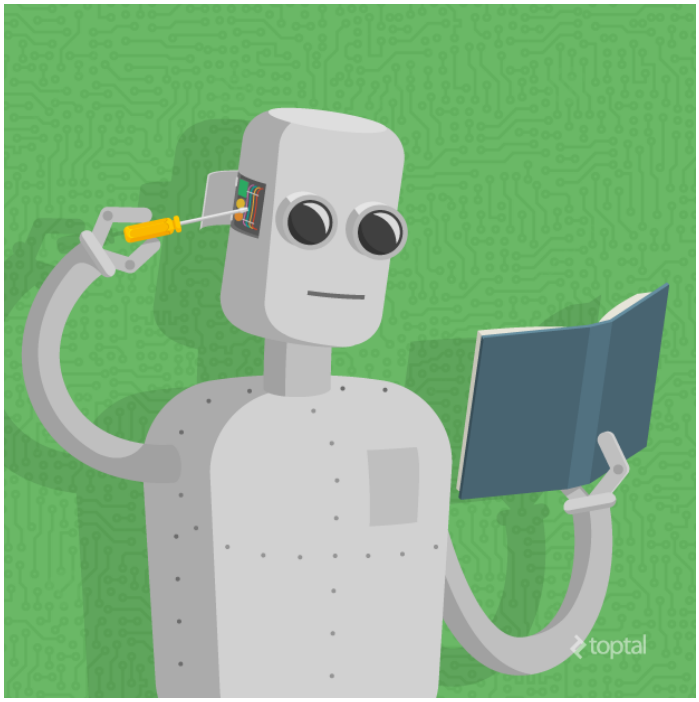
Practical: Machine Learning

BaselRBootcamp 2017



Source: https://www.toptal.com/machine-learning/machine-learning-theory-an-introductory-primer (https://www.toptal.com/machine-learning/machine-learning-theory-an-introductory-primer)

Slides

- Here are the introduction slides for this practical on machine learning! (https://therbootcamp.github.io/_sessions/D2S3_MachineLearning/MachineLearning.html)

Overview

In this practical you'll conduct machine learning analyses on a dataset on heart disease. You will see how well many different machine learning models can predict new data. By the end of this practical you will know how to:

1. Create separate training and test data
2. Fit a model to data
3. Make predictions from a model
4. Compare models in how well they can predict new data.

Glossary and packages

Here are the main functions and packages you'll be using. For more information about the specific models, click on the link in *Additional Details*.

Algorithm	Function	Package	Additional Details
Regression	`glm()`	Base R	https://bookdown.org/ndphillips/YaRrr/regression.html#the-linear-model (https://bookdown.org/ndphillips/YaRrr/regression.html#the-linear-model)
Fast-and-Frugal trees	`FFTrees()`	FFTrees	https://cran.r-project.org/web/packages/FFTrees/vignettes/guide.html (https://cran.r-project.org/web/packages/FFTrees/vignettes/guide.html)

Algorithm	Function	Package	Additional Details
Support Vector Machines	`svm()`	`e1071`	https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html (https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html)
Decision Trees	`rpart()`	`rpart`	https://statweb.stanford.edu/~lpekelis/talks/13_datafest_cart_talk.pdf (https://statweb.stanford.edu/~lpekelis/talks/13_datafest_cart_talk.pdf)
Random Forests	`randomForest()`	`randomForest`	http://www.blopig.com/blog/2017/04/a-very-basic-introduction-to-random-forests-using-r/ (http://www.blopig.com/blog/2017/04/a-very-basic-introduction-to-random-forests-using-r/)

Examples

- The following examples will take you through all steps of the machine learning process, from creating training and test data, to fitting models, to making predictions. Follow along and try to see how piece of code works!

```r
# ------------------------------------------------
# A step-by-step tutorial for conducting machine learning
# In this tutorial, we'll see how well 3 different models can
#  predict medical data
# ------------------------------------------------

# ------------------------
# Part A:
# Load libraries
# ------------------------

library(e1071)          # for svm()
library(randomForest)   # for randomForest()
library(rpart)          # for rpart()
library(yarrr)          # for pirateplot()
library(tidyverse)      # for datawrangling and ggplot2
library(FFTrees)        # for the heartdisease data

# ------------------------
# Part B: Create datasets
#  heart_train, heart_test
#  heart_train_fac, heart_test_fac
# ------------------------

heart <- heartdisease   # Save a copy of the heartdisease data as heart

set.seed(101)   # To fix the training / test randomization

# Randomly sort rows
heart <- heart %>%
  arrange(rnorm(nrow(heart)))

# Savew first 125 rows as heart_train and remaining as heart_test
heart_train <- heart %>% slice(1:100)
heart_test <- heart %>% slice(101:nrow(heart))


# Create heart_train_fact, heart_test_fact
#  Just heart_train and hear_test with factors
#  We're only doing this because the randomForest() function
#   requires factors!!!!

heart_train_fac <- heart_train
heart_test_fac <- heart_test

for(i in 1:ncol(heart_train_fac)) {  # Convert character columns and diagnosis to factor

  if(class(heart_train_fac[[i]]) == "character") {

    heart_train_fac[[i]] <- factor(heart_train_fac[[i]])
    heart_test_fac[[i]] <- factor(heart_test_fac[[i]])

  }}


# ------------------------------
# Part I: Build Models
# ------------------------------

# Build FFTrees_model
FFTrees_model <- FFTrees(formula = sex ~ .,
                         data = heart_train)
```

```
Growing FFTs with ifan
```

```
Fitting non-FFTrees algorithms for comparison (you can turn this off with do.comp = FALSE)
 ...
```

```
# Build glm_model
glm_model <- glm(formula = factor(sex) ~ .,
                 data = heart_train,
                 family = "binomial")  # For predicting a binary variable

# Build randomForest model
randomForest_model <- randomForest(formula = factor(sex) ~ .,
                                   data = heart_train_fac)


# ------------------------------
# Part II: Explore Models
# ------------------------------

print(FFTrees_model)
```

```
FFT #1 predicts sex using 3 cues: {thal,diagnosis,chol}

[1] If thal = {rd,fd}, predict True.
[2] If diagnosis <= 0, predict False.
[3] If chol >= 274, predict False, otherwise, predict True.

                   train
cases       :n    100.00
speed       :mcu    1.76
frugality   :pci    0.87
accuracy    :acc    0.65
weighted    :wacc   0.68
sensitivity :sens   0.56
specificity :spec   0.81

pars: algorithm = 'ifan', goal = 'wacc', goal.chase = 'bacc', sens.w = 0.5, max.levels = 4
```

```
summary(FFTrees_model)
```

```
$train
  tree   n hi mi fa cr      sens      spec       ppv       npv       far
1    1 100 35 28  7 30 0.5555556 0.8108108 0.8333333 0.5172414 0.1891892
2    2 100 43 20 14 23 0.6825397 0.6216216 0.7543860 0.5348837 0.3783784
3    3 100 22 41  4 33 0.3492063 0.8918919 0.8461538 0.4459459 0.1081081
4    4 100 21 42  4 33 0.3333333 0.8918919 0.8400000 0.4400000 0.1081081
5    5 100 13 50  0 37 0.2063492 1.0000000 1.0000000 0.4252874 0.0000000
6    6 100 61  2 29  8 0.9682540 0.2162162 0.6777778 0.8000000 0.7837838
   acc      bacc      wacc       bpv    dprime cost       pci  mcu
1 0.65 0.6831832 0.6831832 0.6752874 1.0205984 0.35 0.8742857 1.76
2 0.66 0.6520807 0.6520807 0.6446348 0.7845550 0.34 0.8135714 2.61
3 0.55 0.6205491 0.6205491 0.6460499 0.8491882 0.45 0.9035714 1.35
4 0.54 0.6126126 0.6126126 0.6400000 0.8059249 0.46 0.8778571 1.71
5 0.50 0.6031746 0.6031746 0.7126437 1.4183470 0.50 0.8735714 1.77
6 0.69 0.5922351 0.5922351 0.7388889 1.0706939 0.31 0.8350000 2.31

$test
NULL
```

```
print(glm_model)
```

```
Call:  glm(formula = factor(sex) ~ ., family = "binomial", data = heart_train)

Coefficients:
       (Intercept)                 age                cpaa
         26.222011           -0.014514           -0.706110
              cpnp                cpta            trestbps
          0.252985           17.219429           -0.030686
              chol                 fbs  restecghypertrophy
         -0.014895            1.247057           -0.029339
     restecgnormal             thalach                exang
         -0.560743           -0.004358            0.498999
           oldpeak           slopeflat             slopeup
          0.090037            0.947599            3.018078
                ca          thalnormal              thalrd
         -0.358819          -19.510997          -17.336611
         diagnosis
          1.711157

Degrees of Freedom: 99 Total (i.e. Null);  81 Residual
Null Deviance:      131.8
Residual Deviance: 87.29    AIC: 125.3
```

```
summary(glm_model)
```

```
Call:
glm(formula = factor(sex) ~ ., family = "binomial", data = heart_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8712  -0.7659   0.1889   0.7371   2.1677

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)         2.622e+01  3.956e+03   0.007   0.9947
age                -1.451e-02  3.962e-02  -0.366   0.7141
cpaa               -7.061e-01  8.216e-01  -0.859   0.3901
cpnp                2.530e-01  7.606e-01   0.333   0.7394
cpta                1.722e+01  1.382e+03   0.012   0.9901
trestbps           -3.069e-02  1.998e-02  -1.536   0.1246
chol               -1.490e-02  6.050e-03  -2.462   0.0138 *
fbs                 1.247e+00  9.524e-01   1.309   0.1904
restecghypertrophy -2.934e-02  5.595e+03   0.000   1.0000
restecgnormal      -5.607e-01  5.595e+03   0.000   0.9999
thalach            -4.358e-03  1.604e-02  -0.272   0.7858
exang               4.990e-01  7.497e-01   0.666   0.5057
oldpeak             9.004e-02  3.952e-01   0.228   0.8198
slopeflat           9.476e-01  1.338e+00   0.708   0.4790
slopeup             3.018e+00  1.480e+00   2.039   0.0415 *
ca                 -3.588e-01  4.061e-01  -0.884   0.3770
thalnormal         -1.951e+01  3.956e+03  -0.005   0.9961
thalrd             -1.734e+01  3.956e+03  -0.004   0.9965
diagnosis           1.711e+00  1.025e+00   1.669   0.0952 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 131.791  on 99  degrees of freedom
Residual deviance:  87.291  on 81  degrees of freedom
AIC: 125.29

Number of Fisher Scoring iterations: 16
```

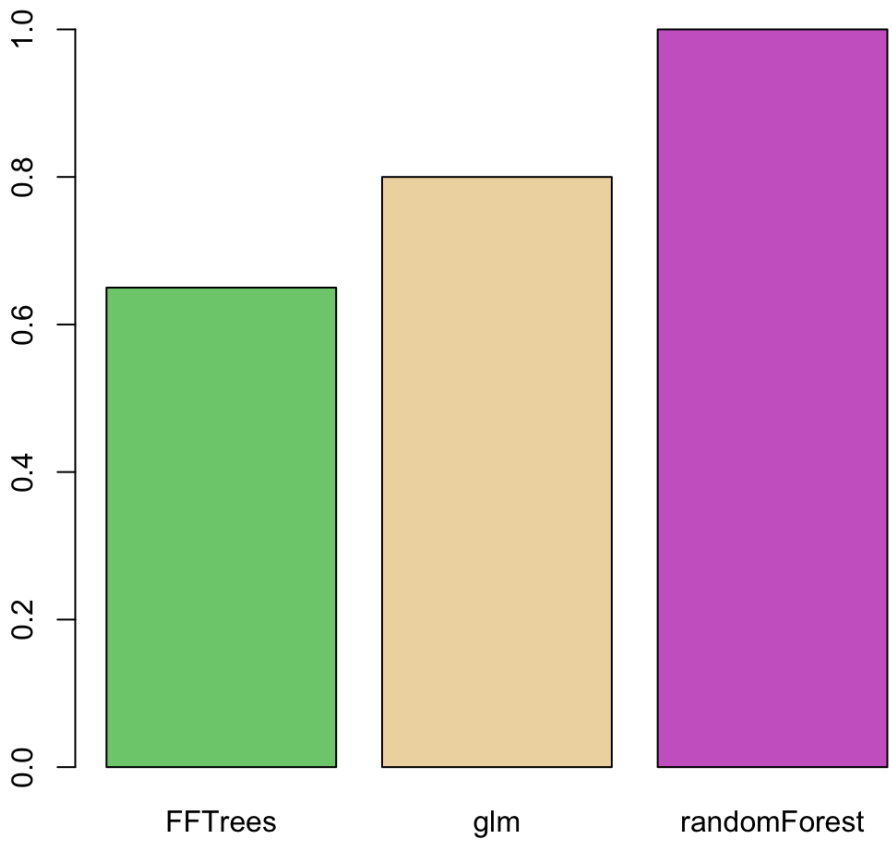
```
print(randomForest_model)
```

```
Call:
 randomForest(formula = factor(sex) ~ ., data = heart_train_fac)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 29%
Confusion matrix:
   0  1 class.error
0 16 21   0.5675676
1  8 55   0.1269841
```

```
summary(randomForest_model)
```

```
               Length Class  Mode
call              3   -none- call
type              1   -none- character
predicted       100   factor numeric
err.rate       1500   -none- numeric
confusion         6   -none- numeric
votes           200   matrix numeric
oob.times       100   -none- numeric
classes           2   -none- character
importance       13   -none- numeric
importanceSD      0   -none- NULL
localImportance   0   -none- NULL
proximity         0   -none- NULL
ntree             1   -none- numeric
mtry              1   -none- numeric
forest           14   -none- list
y               100   factor numeric
test              0   -none- NULL
inbag             0   -none- NULL
terms             3   terms  call
```

```
# ------------------------------
# Part III: Training Accuracy
# ------------------------------

# FFTrees training decisions
FFTrees_fit <- predict(FFTrees_model, heart_train)

# Regression training decisions
#  Positive values are predicted to be 1, negative values are 0
glm_fit <- predict(glm_model, heart_train) > 0

# randomForest training decisions
randomForest_fit <- predict(randomForest_model, heart_train_fac)

# Now calculate fitting accuracies and put in dataframe

# Truth value for training data is heart_train$sex
train_truth <- heart_train$sex

# Put training results together
training_results <- data_frame(FFTrees = mean(FFTrees_fit == train_truth),
                               glm = mean(glm_fit == train_truth),
                               randomForest = mean(randomForest_fit == train_truth))

# Plot training results
barplot(height = unlist(training_results),
        main = "Training Results",
        ylim = c(0, 1),
        col = c("palegreen3", "wheat2", "orchid3"))
```


Training Results
1.0
0.8
0.6
0.4
0.2
0.0
FFTrees
glm
randomForest

```r
# ------------------------------
# Part IV: Prediction Accuacy!
# ------------------------------

# Calculate predictions for each model for heart_test

# FFTrees testing decisions
FFTrees_pred <- predict(FFTrees_model, heart_test)

# Regression testing decisions
#  Positive values are predicted to be 1, negative values are 0
glm_pred <- predict(glm_model, heart_test) >= 0

# randomForest testing decisions
randomForest_pred <- predict(randomForest_model, heart_test_fac)

# Now calculate testing accuracies and put in dataframe

# Truth value for test data is heart_test$sex
test_truth <- heart_test$sex

testing_results <- data_frame(FFTrees = mean(FFTrees_pred == test_truth),
                              glm = mean(glm_pred == test_truth),
                              randomForest = mean(randomForest_pred == test_truth))

# Plot testing results
barplot(height = unlist(testing_results),
        main = "Testing Results",
        ylim = c(0, 1),
        col = c("palegreen3", "wheat2", "orchid3"))
```

Tasks

- Note, most of this practical will be copying and pasting code from the Examples and only making small changes.
- You should start by copying and pasting all of the code in the examples into a new .R file.
- Try running pieces of the code line by line and understand what it's doing!

Part A: Load packages

A. Load all of the necessary packages. For this practical, we'll need `FFTrees`, `e1071`, `randomForest`, `rpart`, and `tidyverse`.

Part B: Create training and test data

B. Now run the code in Part B to save a copy of the data as a tibble called `heart`. Afterwards, print it to make sure it looks ok.

C. Create separate training dataframes `heart_train` (and `heart_train_fac`) for model training and `heart_test` (and `heart_test_fac`) for model testing. Print each of these dataframes to make sure they look ok.

Part I: Train models on `diagnosis`

1. In our analyses, we will try to predict each patient's diagnosis `diagnosis`. Look at the help menu for `heartdisease` to see what this, and the other variables, mean. Then, Create three new model objects `FFTrees_model`, `glm_model`, and `randomForest_model`, each predicting `diagnosis`.

```
# Build FFTrees_model
FFTrees_model <- FFTrees(formula = diagnosis ~ .,
                         data = heart_train)
```

```
Growing FFTs with ifan
```

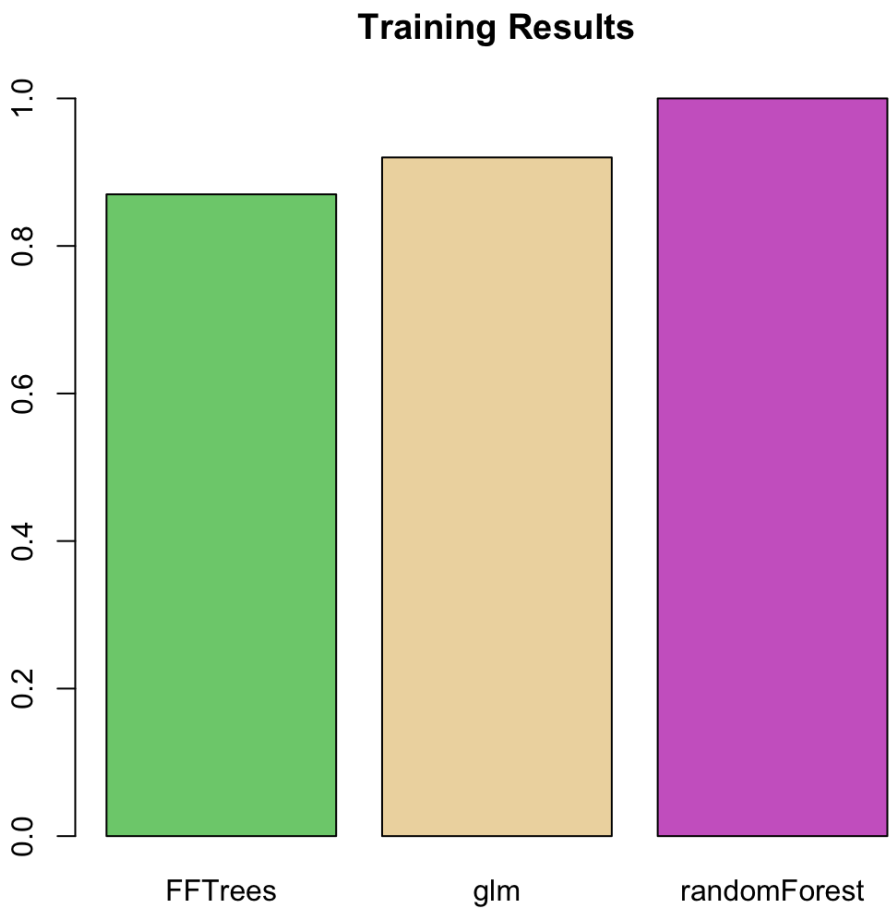
```
Fitting non-FFTrees algorithms for comparison (you can turn this off with do.comp = FALSE)
 ...
```

```
# Build glm_model
glm_model <- glm(formula = factor(diagnosis) ~ .,
                 data = heart_train,
                 family = "binomial")  # For predicting a binary variable

# Build randomForest model
randomForest_model <- randomForest(formula = factor(diagnosis) ~ .,
                                   data = heart_train_fac)
```

Part II: Calculate fits for training data

2. Calculate fits for each model with `predict()`, then create `training_results` containing the fitting accuracy of each model in a dataframe. The code will be almost identical to what is in the Example. All you need to do is change the value of `truth_train` to the correct column in `heart_train`. Afterwards, plot the results using `barplot()`. Which model had the best training accuracy?

```
# FFTrees training decisions
FFTrees_fit <- predict(FFTrees_model, heart_train)

# Regression training decisions
#  Positive values are predicted to be 1, negative values are 0
glm_fit <- predict(glm_model, heart_train) > 0

# randomForest training decisions
randomForest_fit <- predict(randomForest_model, heart_train_fac)

# Now calculate fitting accuracies and put in dataframe

# Truth value for training data is heart_train$sex
train_truth <- heart_train$diagnosis

# Put training results together
training_results <- data_frame(FFTrees = mean(FFTrees_fit == train_truth),
                               glm = mean(glm_fit == train_truth),
                               randomForest = mean(randomForest_fit == train_truth))

# Plot training results
barplot(height = unlist(training_results),
        main = "Training Results",
        ylim = c(0, 1),
        col = c("palegreen3", "wheat2", "orchid3"))
```



Part III: Explore models

3. Explore each of the three models by applying `print()` and `summary()`. Can you interpret any of them?

```
print(FFTrees_model)
```

```
FFT #1 predicts diagnosis using 3 cues: {cp,thal,ca}

[1] If cp != {a}, predict False.
[2] If thal = {rd,fd}, predict True.
[3] If ca <= 0, predict False, otherwise, predict True.

                   train
cases       :n    100.00
speed       :mcu    1.57
frugality   :pci    0.89
accuracy    :acc    0.87
weighted    :wacc   0.84
sensitivity :sens   0.70
specificity :spec   0.97

pars: algorithm = 'ifan', goal = 'wacc', goal.chase = 'bacc', sens.w = 0.5, max.levels = 4
```

```
summary(FFTrees_model)
```

```
$train
  tree   n hi mi fa cr      sens      spec       ppv       npv        far
1    1 100 26 11  2 61 0.7027027 0.9682540 0.9285714 0.8472222 0.03174603
2    2 100 26 11  2 61 0.7027027 0.9682540 0.9285714 0.8472222 0.03174603
3    3 100 31  6 12 51 0.8378378 0.8095238 0.7209302 0.8947368 0.19047619
4    4 100 30  7 12 51 0.8108108 0.8095238 0.7142857 0.8793103 0.19047619
5    5 100 33  4 19 44 0.8918919 0.6984127 0.6346154 0.9166667 0.30158730
6    6 100 37  0 30 33 1.0000000 0.5238095 0.5522388 1.0000000 0.47619048
7    7 100 17 20  0 63 0.4594595 1.0000000 1.0000000 0.7590361 0.00000000
8    8 100 11 26  0 63 0.2972973 1.0000000 1.0000000 0.7078652 0.00000000
   acc      bacc      wacc       bpv   dprime cost       pci  mcu
1 0.87 0.8354783 0.8354783 0.8878968 2.387920 0.13 0.8878571 1.57
2 0.87 0.8354783 0.8354783 0.8878968 2.387920 0.13 0.8792857 1.69
3 0.82 0.8236808 0.8236808 0.8078335 1.861753 0.18 0.8700000 1.82
4 0.81 0.8101673 0.8101673 0.7967980 1.757031 0.19 0.8771429 1.72
5 0.77 0.7951523 0.7951523 0.7756410 1.756493 0.23 0.8414286 2.22
6 0.70 0.7619048 0.7619048 0.7761194 2.280303 0.30 0.8271429 2.42
7 0.80 0.7297297 0.7297297 0.8795181 2.318451 0.20 0.8835714 1.63
8 0.74 0.6486486 0.6486486 0.8539326 1.900712 0.26 0.8714286 1.80

$test
NULL
```

```
print(glm_model)
```

```

Call:  glm(formula = factor(diagnosis) ~ ., family = "binomial", data = heart_train)

Coefficients:
       (Intercept)                 age                 sex
         -23.835844            0.237509            1.997761
              cpaa                cpnp                cpta
          -4.839762           -8.253433           -6.099230
           trestbps                chol                 fbs
          -0.021537           -0.005529            0.400766
restecghypertrophy       restecgnormal             thalach
           9.314383            6.031715            0.015270
              exang             oldpeak           slopeflat
          -1.557737            4.037985           -4.643556
            slopeup                  ca          thalnormal
          -3.145630            3.312387            3.331373
             thalrd
           7.443504

Degrees of Freedom: 99 Total (i.e. Null);  81 Residual
Null Deviance:      131.8
Residual Deviance: 29.22    AIC: 67.22
```

```
summary(glm_model)
```

```
Call:
glm(formula = factor(diagnosis) ~ ., family = "binomial", data = heart_train)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.74786  -0.10364  -0.00404   0.00334   2.01591

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)        -2.384e+01  3.956e+03  -0.006   0.9952
age                 2.375e-01  1.579e-01   1.504   0.1326
sex                 1.998e+00  1.978e+00   1.010   0.3126
cpaa               -4.840e+00  2.438e+00  -1.985   0.0471 *
cpnp               -8.253e+00  3.445e+00  -2.396   0.0166 *
cpta               -6.099e+00  2.594e+00  -2.351   0.0187 *
trestbps           -2.154e-02  5.768e-02  -0.373   0.7089
chol               -5.529e-03  1.115e-02  -0.496   0.6199
fbs                 4.008e-01  1.948e+00   0.206   0.8370
restecghypertrophy  9.314e+00  5.595e+03   0.002   0.9987
restecgnormal       6.032e+00  5.595e+03   0.001   0.9991
thalach             1.527e-02  2.765e-02   0.552   0.5808
exang              -1.558e+00  2.257e+00  -0.690   0.4900
oldpeak             4.038e+00  1.643e+00   2.458   0.0140 *
slopeflat          -4.644e+00  3.032e+00  -1.532   0.1256
slopeup            -3.146e+00  3.224e+00  -0.976   0.3293
ca                  3.312e+00  1.233e+00   2.688   0.0072 **
thalnormal          3.331e+00  3.956e+03   0.001   0.9993
thalrd              7.444e+00  3.956e+03   0.002   0.9985
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 131.791  on 99  degrees of freedom
Residual deviance:  29.219  on 81  degrees of freedom
AIC: 67.219

Number of Fisher Scoring iterations: 16
```

```
print(randomForest_model)
```

```
Call:
 randomForest(formula = factor(diagnosis) ~ ., data = heart_train_fac)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 14%
Confusion matrix:
   0  1 class.error
0 60  3  0.04761905
1 11 26  0.29729730
```

```
summary(randomForest_model)
```

```
               Length Class  Mode
call              3   -none- call
type              1   -none- character
predicted       100   factor numeric
err.rate       1500   -none- numeric
confusion         6   -none- numeric
votes           200   matrix numeric
oob.times       100   -none- numeric
classes           2   -none- character
importance       13   -none- numeric
importanceSD      0   -none- NULL
localImportance   0   -none- NULL
proximity         0   -none- NULL
ntree             1   -none- numeric
mtry              1   -none- numeric
forest           14   -none- list
y               100   factor numeric
test              0   -none- NULL
inbag             0   -none- NULL
terms             3   terms  call
```

Part IV: Calculate predictions for test data

4. Calculate predictions for each model based on `heart_test`, and then calculate the prediction accuracies. Don't forget to change the value of `truth_test` to reflect the true value for the current analysis! Then plot the results. Which model predicted each patient's diagnosis the best?

```
# ------------------------------
# Part IV: Prediction Accuacy!
# ------------------------------

# Calculate predictions for each model for heart_test

# FFTrees testing decisions
FFTrees_pred <- predict(FFTrees_model, heart_test)

# Regression testing decisions
#  Positive values are predicted to be 1, negative values are 0
glm_pred <- predict(glm_model, heart_test) >= 0

# randomForest testing decisions
randomForest_pred <- predict(randomForest_model, heart_test_fac)

# Now calculate testing accuracies and put in dataframe

# Truth value for test data is heart_test$sex
test_truth <- heart_test$diagnosis

testing_results <- data_frame(FFTrees = mean(FFTrees_pred == test_truth),
                              glm = mean(glm_pred == test_truth),
                              randomForest = mean(randomForest_pred == test_truth))

# Plot testing results
barplot(height = unlist(testing_results),
        main = "Testing Results",
        ylim = c(0, 1),
        col = c("palegreen3", "wheat2", "orchid3"))
```

Extras and Challenges

5. A fellow colleague things that support vector machines should perform better than the models you used. Is she right? Test her prediction by including support vector machines using the `svm()` function from the `e1071` package in all of your analyses. You'll need to add code for support vector machines at each stage of the machine learning process, model building, data fitting, and data prediction. Was she right?

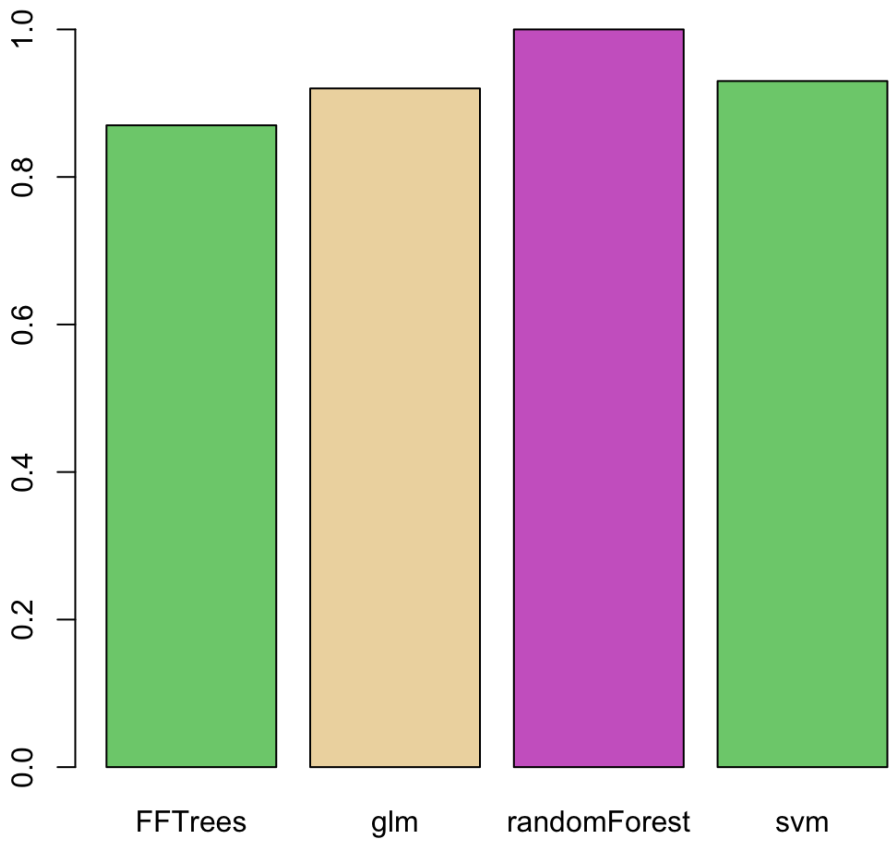
```
# Build svm model
svm_model <- svm(formula = factor(diagnosis) ~ .,
                 data = heart_train_fac)

print(svm_model)
```

```

Call:
svm(formula = factor(diagnosis) ~ ., data = heart_train_fac)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.05263158

Number of Support Vectors:  56
```

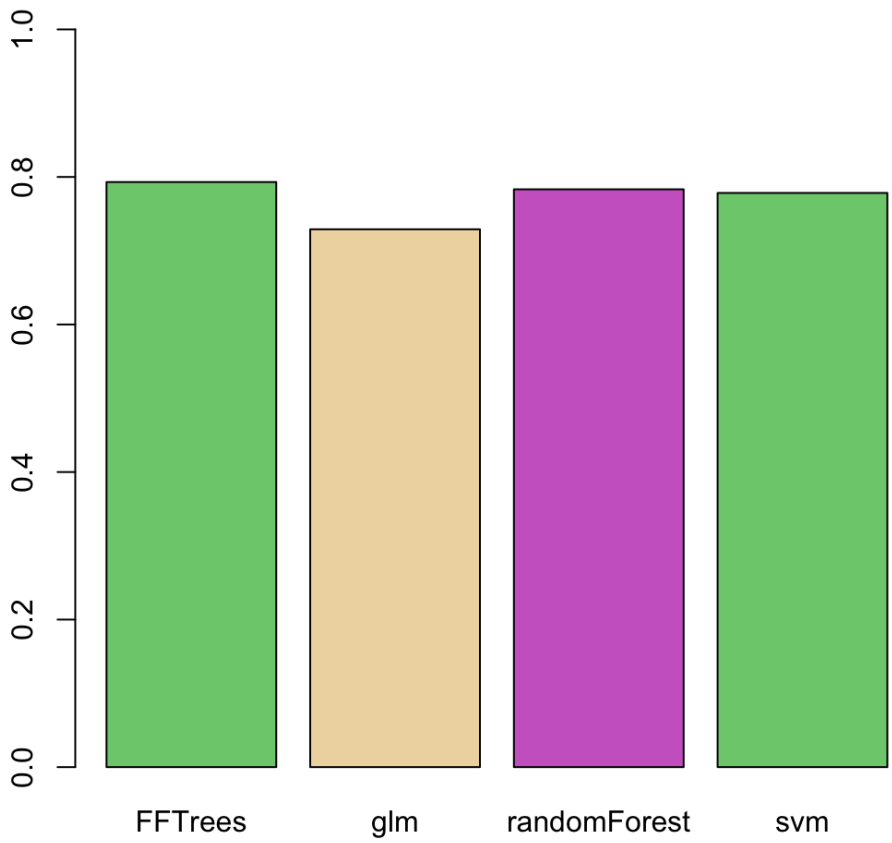
```
summary(svm_model)
```

```
Call:
svm(formula = factor(diagnosis) ~ ., data = heart_train_fac)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.05263158

Number of Support Vectors:  56

 ( 27 29 )


Number of Classes:  2

Levels:
 0 1
```

```
# svm training decisions
svm_fit <- predict(svm_model, heart_train_fac)

# Now calculate fitting accuracies and put in dataframe

# Truth value for training data is heart_train$sex
train_truth <- heart_train$diagnosis

# Put training results together
training_results <- data_frame(FFTrees = mean(FFTrees_fit == train_truth),
                               glm = mean(glm_fit == train_truth),
                               randomForest = mean(randomForest_fit == train_truth),
                               svm = mean(svm_fit == train_truth))

# Plot training results
barplot(height = unlist(training_results),
        main = "Training Results",
        ylim = c(0, 1),
        col = c("palegreen3", "wheat2", "orchid3"))
```




```
# svm testing decisions
svm_pred <- predict(svm_model, heart_test_fac)

# Now calculate testing accuracies and put in dataframe

# Truth value for test data is heart_test$sex
test_truth <- heart_test$diagnosis

testing_results <- data_frame(FFTrees = mean(FFTrees_pred == test_truth),
                              glm = mean(glm_pred == test_truth),
                              randomForest = mean(randomForest_pred == test_truth),
                              svm = mean(svm_pred == test_truth))

# Plot testing results
barplot(height = unlist(testing_results),
        main = "Testing Results",
        ylim = c(0, 1),
        col = c("palegreen3", "wheat2", "orchid3"))
```



6. You'll notice in Part C that we trained the models on 125 cases (out of the 303) in the full dataset. In other words, we trained the data on about half of the total cases. Try repeating the same machine learning process as above (for either cholesterol or resting heart rate), but instead of training the models on 125 cases, try training it on only 50 cases (about 15% of the data). How do you think having fewer cases in the training data will affect accuracy in fitting and prediction? When you are done, try training the models based on 250 cases (over 80% of the data) and then making predictions on the remaining cases.

```
# Savew first 125 rows as heart_train and remaining as heart_test
heart_train <- heart %>% slice(1:250)
heart_test <- heart %>% slice(251:nrow(heart))

# Create heart_train_fact, heart_test_fact
#  Just heart_train and hear_test with factors
#  We're only doing this because the randomForest() function
#   requires factors!!!!

heart_train_fac <- heart_train
heart_test_fac <- heart_test

for(i in 1:ncol(heart_train_fac)) {  # Convert character columns and diagnosis to factor

  if(class(heart_train_fac[[i]]) == "character") {

    heart_train_fac[[i]] <- factor(heart_train_fac[[i]])
    heart_test_fac[[i]] <- factor(heart_test_fac[[i]])

  }}


# ------------------------------
# Part I: Build Models
# ------------------------------

# Build FFTrees_model
FFTrees_model <- FFTrees(formula = sex ~ .,
                         data = heart_train)
```

```
Growing FFTs with ifan
```

```
Fitting non-FFTrees algorithms for comparison (you can turn this off with do.comp = FALSE)
 ...
```

```
# Build glm_model
glm_model <- glm(formula = factor(sex) ~ .,
                 data = heart_train,
                 family = "binomial")  # For predicting a binary variable

# Build randomForest model
randomForest_model <- randomForest(formula = factor(sex) ~ .,
                                   data = heart_train_fac)


# ------------------------------
# Part II: Explore Models
# ------------------------------

print(FFTrees_model)
```

```
FFT #1 predicts sex using 3 cues: {thal,diagnosis,chol}

[1] If thal = {rd,fd}, predict True.
[2] If diagnosis <= 0, predict False.
[3] If chol >= 263, predict False, otherwise, predict True.

                   train
cases       :n    250.00
speed       :mcu    1.69
frugality   :pci    0.88
accuracy    :acc    0.69
weighted    :wacc   0.71
sensitivity :sens   0.66
specificity :spec   0.77

pars: algorithm = 'ifan', goal = 'wacc', goal.chase = 'bacc', sens.w = 0.5, max.levels = 4
```

```
summary(FFTrees_model)
```

```
$train
  tree   n  hi  mi fa cr      sens      spec       ppv       npv
1    1 250 113  59 18 60 0.6569767 0.7692308 0.8625954 0.5042017
2    2 250  93  79 12 66 0.5406977 0.8461538 0.8857143 0.4551724
3    3 250 161  11 49 29 0.9360465 0.3717949 0.7666667 0.7250000
4    4 250  63 109  9 69 0.3662791 0.8846154 0.8750000 0.3876404
5    5 250  32 140  3 75 0.1860465 0.9615385 0.9142857 0.3488372
6    6 250 172   0 70  8 1.0000000 0.1025641 0.7107438 1.0000000
         far   acc      bacc      wacc       bpv    dprime  cost       pci
1 0.23076923 0.692 0.7131038 0.7131038 0.6833986 1.1405420 0.308 0.8794286
2 0.15384615 0.636 0.6934258 0.6934258 0.6704433 1.1222678 0.364 0.8871429
3 0.62820513 0.760 0.6539207 0.6539207 0.7458333 1.1953043 0.240 0.8388571
4 0.11538462 0.528 0.6254472 0.6254472 0.6313202 0.8566551 0.472 0.8645714
5 0.03846154 0.428 0.5737925 0.5737925 0.6315615 0.8762654 0.572 0.8585714
6 0.89743590 0.720 0.5512821 0.5512821 0.8553719 1.5205678 0.280 0.8525714
    mcu
1 1.688
2 1.580
3 2.256
4 1.896
5 1.980
6 2.064

$test
NULL
```

```
print(glm_model)
```

```
Call:  glm(formula = factor(sex) ~ ., family = "binomial", data = heart_train)

Coefficients:
       (Intercept)                 age                cpaa
         6.3988676          -0.0401754           0.6502277
              cpnp                cpta            trestbps
         0.6264185           2.6360518          -0.0246301
              chol                 fbs  restecghypertrophy
        -0.0095315           0.4014602           3.2831441
     restecgnormal             thalach               exang
         2.8096276           0.0001965           0.6858391
          oldpeak           slopeflat             slopeup
         0.1849547          -0.0088164           1.3117435
               ca           thalnormal              thalrd
        -0.0573891          -3.5801233          -1.5789970
         diagnosis
         1.2653622

Degrees of Freedom: 249 Total (i.e. Null);  231 Residual
Null Deviance:      310.3
Residual Deviance: 226.1    AIC: 264.1
```

```
summary(glm_model)
```

```
Call:
glm(formula = factor(sex) ~ ., family = "binomial", data = heart_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7633  -0.8280   0.3354   0.7766   1.8761

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)         6.3988676  2.8750710   2.226  0.02604 *
age                -0.0401754  0.0237256  -1.693  0.09039 .
cpaa                0.6502277  0.5280215   1.231  0.21816
cpnp                0.6264185  0.4646348   1.348  0.17760
cpta                2.6360518  0.9418463   2.799  0.00513 **
trestbps           -0.0246301  0.0108526  -2.270  0.02324 *
chol               -0.0095315  0.0034554  -2.758  0.00581 **
fbs                 0.4014602  0.4891237   0.821  0.41177
restecghypertrophy  3.2831441  1.6103716   2.039  0.04148 *
restecgnormal       2.8096276  1.6081684   1.747  0.08062 .
thalach             0.0001965  0.0094637   0.021  0.98343
exang               0.6858391  0.4668781   1.469  0.14184
oldpeak             0.1849547  0.2144150   0.863  0.38836
slopeflat          -0.0088164  0.7415542  -0.012  0.99051
slopeup             1.3117435  0.8408030   1.560  0.11873
ca                 -0.0573891  0.2203662  -0.260  0.79454
thalnormal         -3.5801233  1.3170325  -2.718  0.00656 **
thalrd             -1.5789970  1.3139983  -1.202  0.22949
diagnosis           1.2653622  0.5195705   2.435  0.01488 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 310.35  on 249  degrees of freedom
Residual deviance: 226.08  on 231  degrees of freedom
AIC: 264.08

Number of Fisher Scoring iterations: 6
```

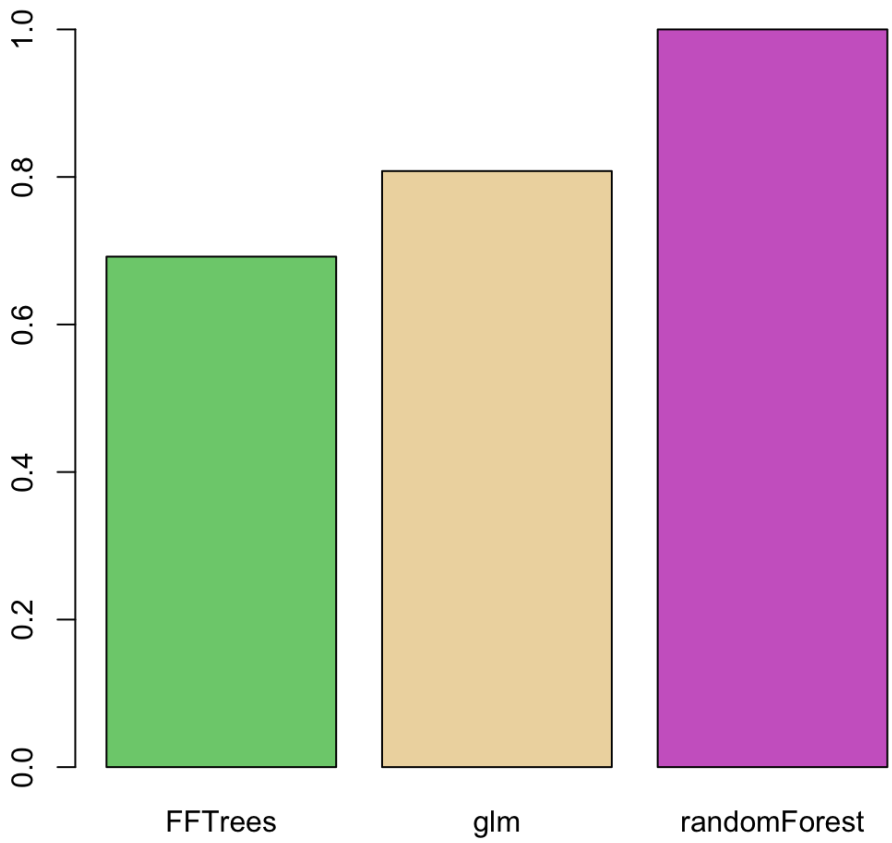
```
print(randomForest_model)
```

```
Call:
 randomForest(formula = factor(sex) ~ ., data = heart_train_fac)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 24%
Confusion matrix:
   0   1 class.error
0 34  44  0.56410256
1 16 156  0.09302326
```

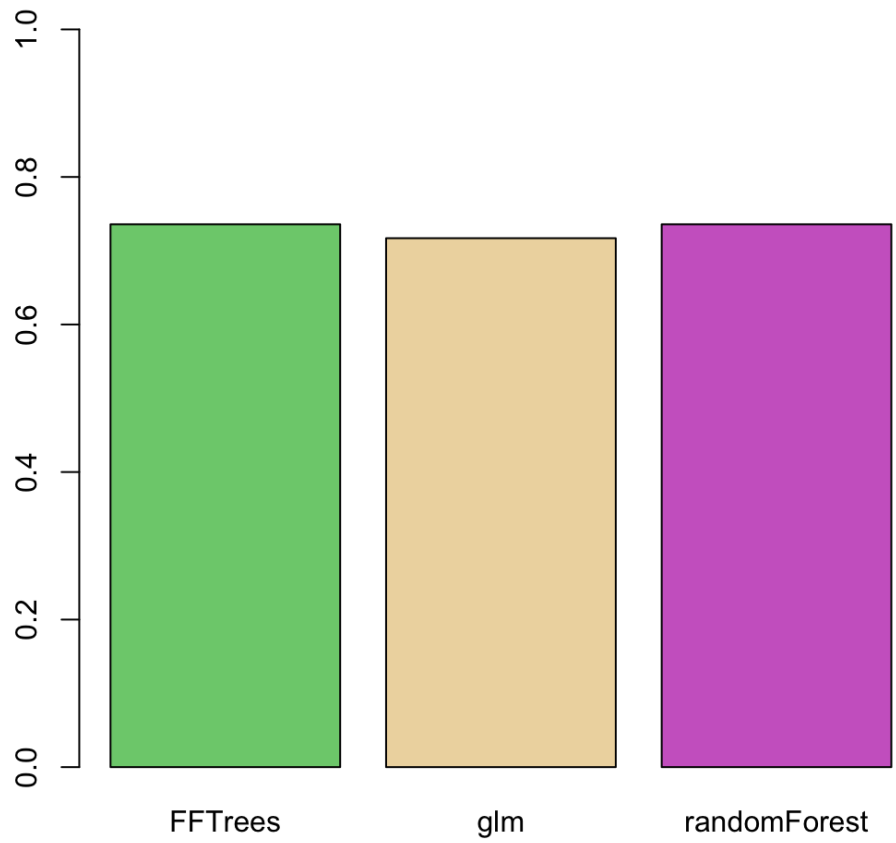
```
summary(randomForest_model)
```

```
               Length Class  Mode
call              3   -none- call
type              1   -none- character
predicted       250   factor numeric
err.rate       1500   -none- numeric
confusion         6   -none- numeric
votes           500   matrix numeric
oob.times       250   -none- numeric
classes           2   -none- character
importance       13   -none- numeric
importanceSD      0   -none- NULL
localImportance   0   -none- NULL
proximity         0   -none- NULL
ntree             1   -none- numeric
mtry              1   -none- numeric
forest           14   -none- list
y               250   factor numeric
test              0   -none- NULL
inbag             0   -none- NULL
terms             3   terms  call
```

```
# ------------------------------
# Part III: Training Accuracy
# ------------------------------

# FFTrees training decisions
FFTrees_fit <- predict(FFTrees_model, heart_train)

# Regression training decisions
#  Positive values are predicted to be 1, negative values are 0
glm_fit <- predict(glm_model, heart_train) > 0

# randomForest training decisions
randomForest_fit <- predict(randomForest_model, heart_train_fac)

# Now calculate fitting accuracies and put in dataframe

# Truth value for training data is heart_train$sex
train_truth <- heart_train$sex

# Put training results together
training_results <- data_frame(FFTrees = mean(FFTrees_fit == train_truth),
                               glm = mean(glm_fit == train_truth),
                               randomForest = mean(randomForest_fit == train_truth))

# Plot training results
barplot(height = unlist(training_results),
        main = "Training Results",
        ylim = c(0, 1),
        col = c("palegreen3", "wheat2", "orchid3"))
```


Training Results
1.0
0.8
0.6
0.4
0.2
0.0
FFTrees
glm
randomForest

```r
# ------------------------------
# Part IV: Prediction Accuacy!
# ------------------------------

# Calculate predictions for each model for heart_test

# FFTrees testing decisions
FFTrees_pred <- predict(FFTrees_model, heart_test)

# Regression testing decisions
#  Positive values are predicted to be 1, negative values are 0
glm_pred <- predict(glm_model, heart_test) >= 0

# randomForest testing decisions
randomForest_pred <- predict(randomForest_model, heart_test_fac)

# Now calculate testing accuracies and put in dataframe

# Truth value for test data is heart_test$sex
test_truth <- heart_test$sex

testing_results <- data_frame(FFTrees = mean(FFTrees_pred == test_truth),
                              glm = mean(glm_pred == test_truth),
                              randomForest = mean(randomForest_pred == test_truth))

# Plot testing results
barplot(height = unlist(testing_results),
        main = "Testing Results",
        ylim = c(0, 1),
        col = c("palegreen3", "wheat2", "orchid3"))
```



7. In all of our machine learning, we have allowed all models to use all data in the `heartdisease` dataset. What do you think would happen if we only let the models use a single predictor like `age`? Test your prediction by replicating the machine learning process, but *only* allow the models to make predictions based on `age`, `cp` and `slope`. (Hint: You can easily tell a model what specific variables to include using the `formula` argument. For example, `formula = y ~ a + b` will tell a model to model a variable y, but *only* using variables a and b.)

```
# Just use formula = diagnosis ~ age + cp + slope in fitting the models.
```

8. How do you think these algorithms would perform on a randomly generated dataset? Let's test this by creating a random training and test dataset, and then see how well the algorithms do. Run the code below to add a random column of data called `random` to `heart_train` and `heart_test`. Then, run your machine learning analysis, but now train and test the models on the new random data column. How well do the models do in training and testing?

```
# Add a new column random to heart_train and heart_test

heart_train$random <- sample(c(0, 1), size = nrow(heart_train), replace = TRUE)

heart_test$random <- sample(c(0, 1), size = nrow(heart_test), replace = TRUE)
```

9. So far we've only looked at the `heartdisease` data. Try conducting a similar analysis on the `ACTG175` data from the `speff2trial` package. For example, you could try to predict whether or not a patient was a intravenous drug user. Which of the different machine learning algorithms performs the best in predicting new data from this dataset? Do you discover any challenges in working with dataset that weren't present in the `heartdisease` data?

```
# Try on your own!
```

Additional reading

- For more advanced machine learning functionality in R, check out the `caret` package caret documentation link (http://topepo.github.io/caret/index.html) and the `mlr` package mlr documentation link (https://cran.r-project.org/web/packages/mlr/vignettes/mlr.html). Both of these packages contain functions that automate most aspects of the machine learning process.
- To read more about the fundamentals of machine learning and statistical prediction, check out An Introduction to Statistical Learning by James et al. (https://www.amazon.com/Introduction-Statistical-Learning-Applications-Statistics/dp/1461471370)