PCOS modeling

Erick Navarro

2023-01-25

Contents

Creating the classificators

This report contains the details of the generation and comparison of several classificators whose aim is to classify people with and without Polycystic ovary syndrome (PCOS). The goal of this project is to develop and validate a model that takes easily collected variables, and test its performance against models that possess variables obtained with increasing invasive proceedures. For a summary of the results and a clearer picture of the project, please read the Final_report.pdf file in this repository.

Setup

Load packages and data

Data splitting

After loading the data and packages, we will proceed to split the data into training and validation sets.

```r
data = data %>%
  column_to_rownames("id")

train.index <- caret::createDataPartition(data$pcos, p = .7, list=FALSE)

train <- data[ train.index,]
valid  <- data[-train.index,]


#Check the negative/positive ration in the validation set
table(train$pcos)
```

```
##
##  No Yes
## 255 124
```

```r
table(train$pcos)[1]/ table(train$pcos)[2]
```

```
##       No
## 2.056452
```

```r
#Check the negative/positive ration in the validation set
table(valid$pcos)
```

```
##
##  No Yes
## 109  53
```

```r
table(valid$pcos)[1]/ table(valid$pcos)[2]
```

```
##       No
## 2.056604
```

```r
#Check the negative/positive ration in the original data set
table(data$pcos)[1]/ table(data$pcos)[2]
```

```
##       No
## 2.056497
```

Imputation on training set

Following the data splitting, we will impute missing values. As we observed in the EDA analysis, the proportion of missing data is overall very low, which makes all the variables with misingness suitable for imputation

```r
# Explore misingness in training set
sapply(train, function(x) sum(is.na(x)))
```

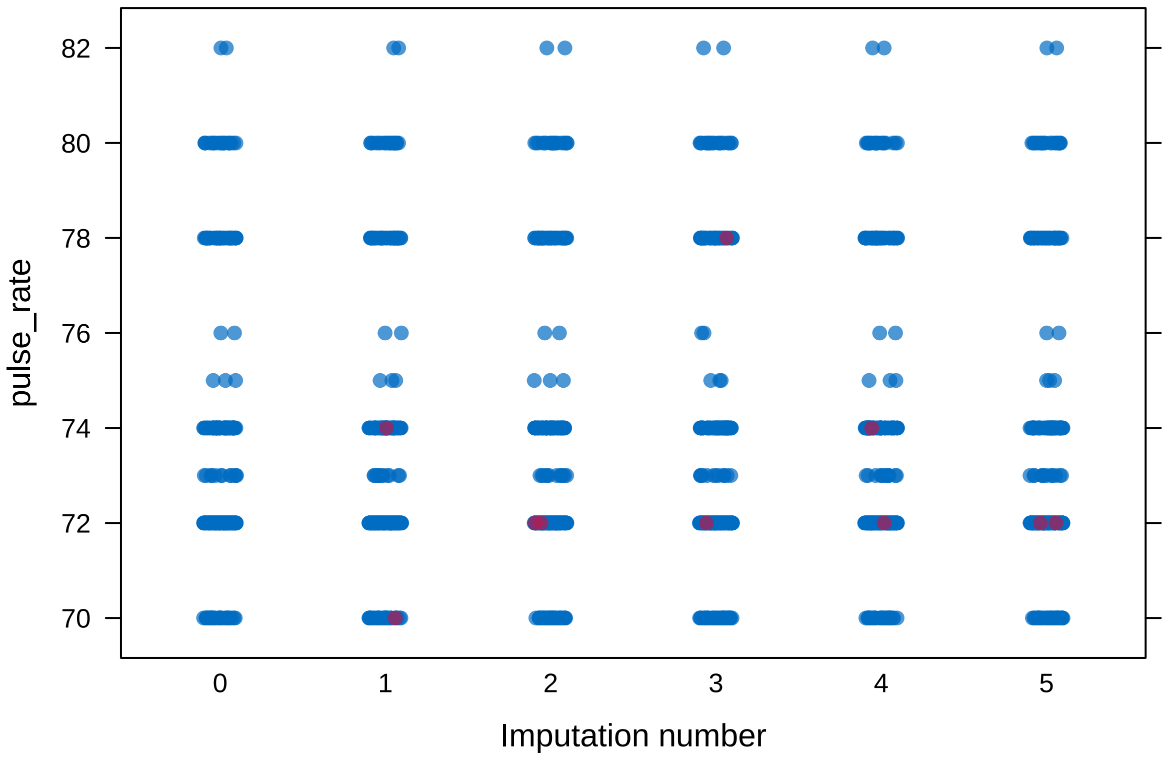
```
##            pcos             age          weight          height             bmi 
##               0               0               0               0               0 
##     blood_group      pulse_rate              rr              hb           cycle 
##               0               2               0               0               0 
##    cycle_length marriage_status        pregnant no_of_abortions       i_betahcg 
##               0               0               0               0               0 
##      ii_betahcg             fsh              lh    fsh_lh_ratio             hip 
##               0               0               1               1               0 
##           waist waist_hip_ratio             tsh             amh             prl 
##               0               0               0               2               0 
##           vitd3             prg             rbs     weight_gain     hair_growth 
##               1               0               0               0               0 
##  skin_darkening       hair_loss         pimples       fast_food    reg_exercise 
##               0               0               0               1               0 
##     bp_systolic    bp_diastolic   follicle_no_l   follicle_no_r    avg_f_size_l 
##               0               0               0               0               0 
##    avg_f_size_r     endometrium 
##               0               0
```

For imputation, we will use the mice R package, a widely used software to handle missing data. We will use the default options, which use the most appropuate methodology depending on the class of the data to be imputed, which are the following: "By default, the method uses pmm, predictive mean matching (numeric data) logreg, logistic regression imputation (binary data, factor with 2 levels) polyreg, polytomous regression imputation for un- ordered categorical data (factor > 2 levels) polr, proportional odds model for (ordered, > 2 levels)."

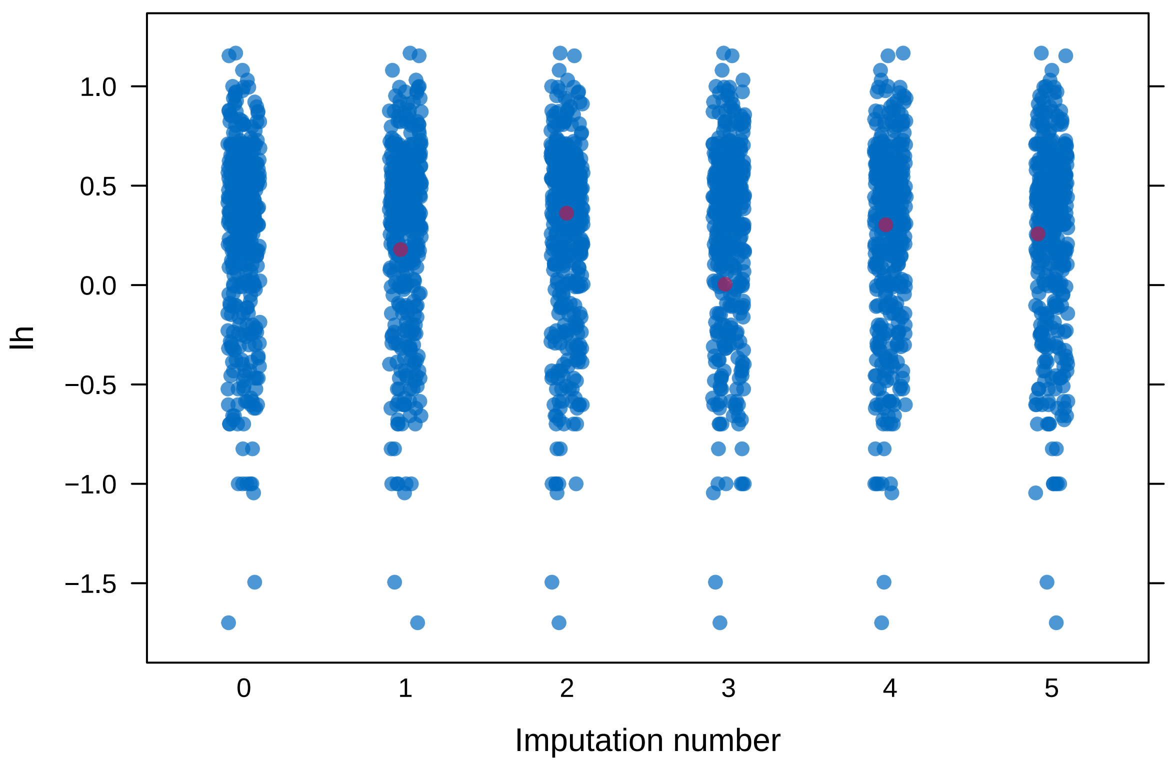
```
chained_train = mice::mice(train)
```

```
## 
##  iter imp variable
##   1   1  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   1   2  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   1   3  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   1   4  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   1   5  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   2   1  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   2   2  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   2   3  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   2   4  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   2   5  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   3   1  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   3   2  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   3   3  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   3   4  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   3   5  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   4   1  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   4   2  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   4   3  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   4   4  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   4   5  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   5   1  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   5   2  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   5   3  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   5   4  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food
##   5   5  pulse_rate  lh  fsh_lh_ratio  amh  vitd3  fast_food

## Warning: Number of logged events: 98
```

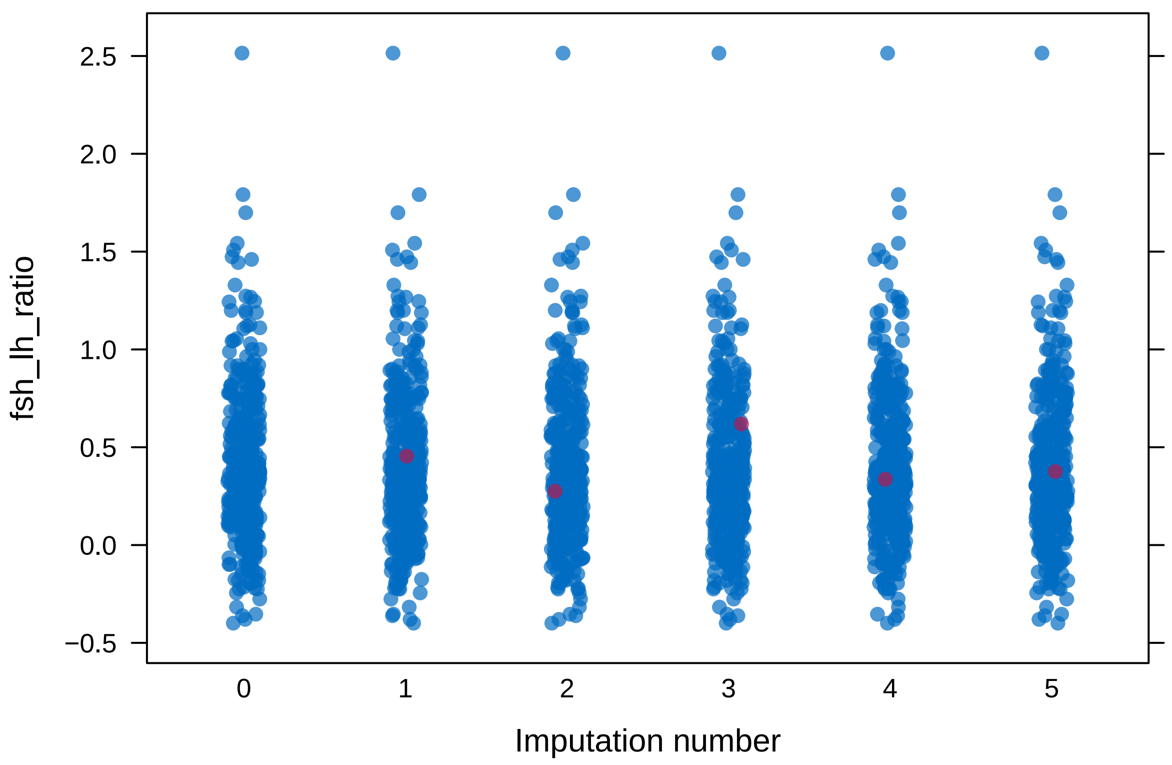
```
#Explore the imputed data and check that the generated value are plausible
stripplot(chained_train, pulse_rate, pch = 19, xlab = "Imputation number")
```



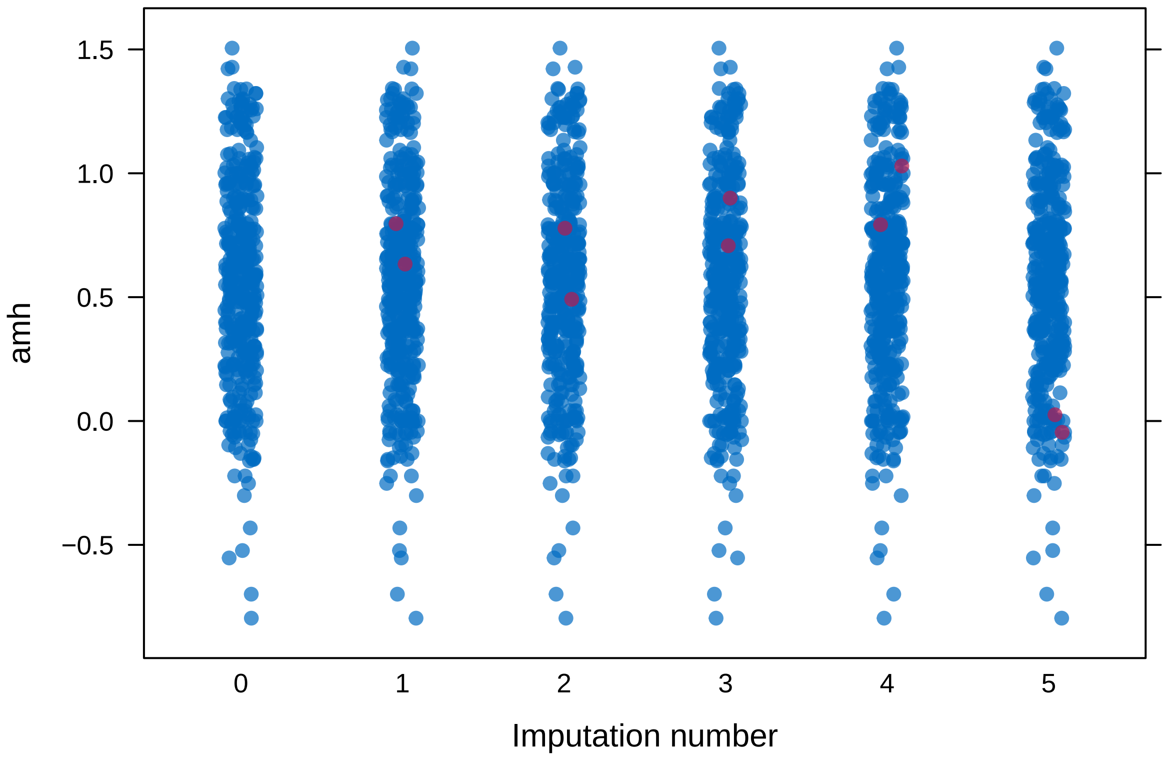
```
stripplot(chained_train, lh, pch = 19, xlab = "Imputation number")
```



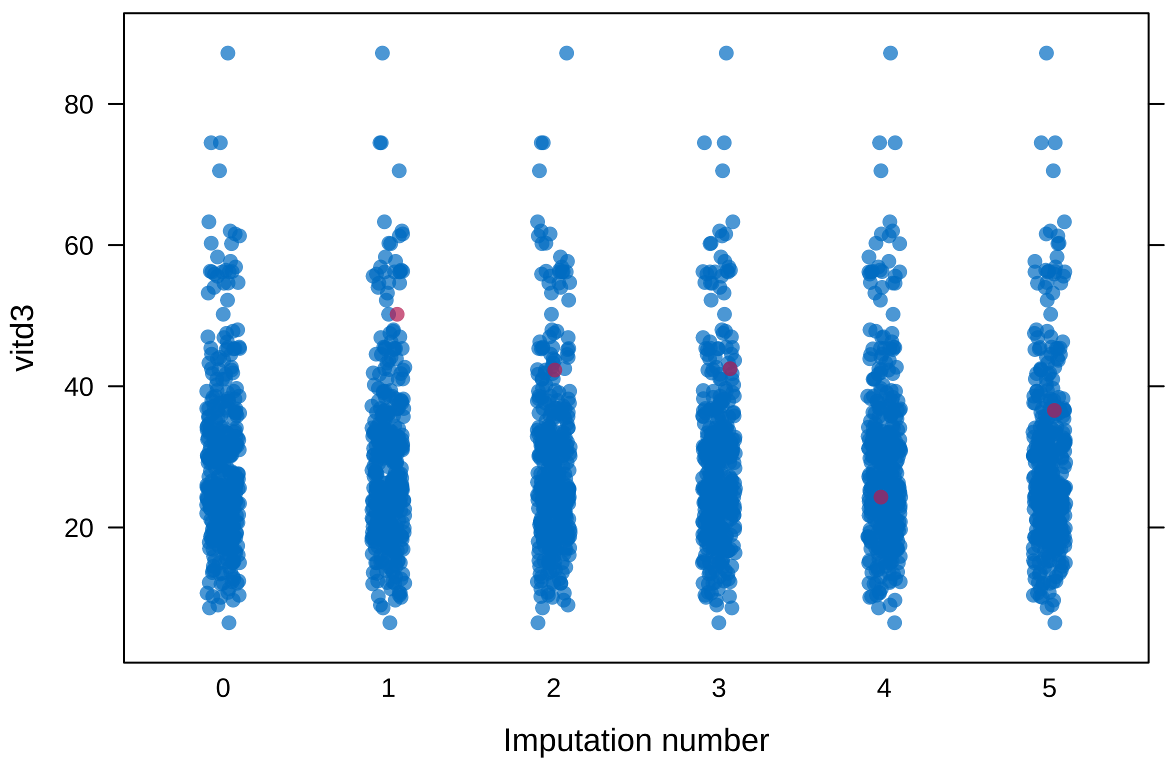
```
stripplot(chained_train, fsh_lh_ratio, pch = 19, xlab = "Imputation number")
```



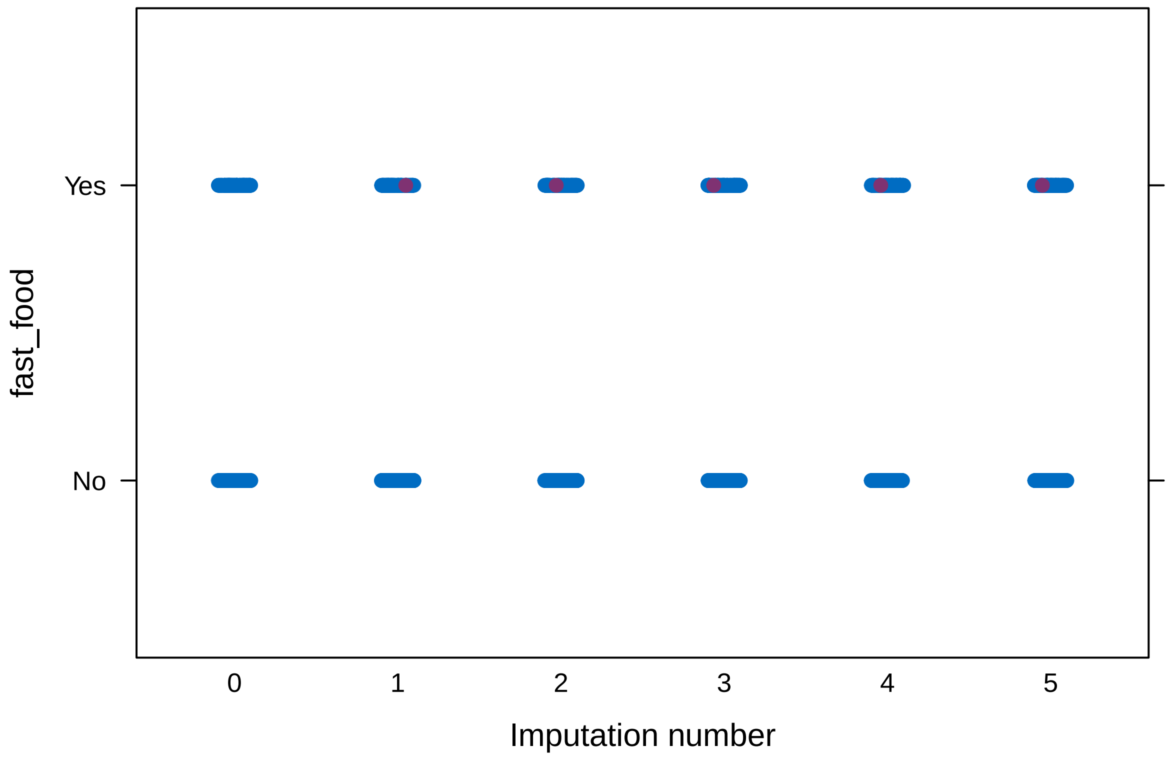
```
stripplot(chained_train, amh, pch = 19, xlab = "Imputation number")
```



```
stripplot(chained_train, vitd3, pch = 19, xlab = "Imputation number")
```



```
stripplot(chained_train, fast_food, pch = 19, xlab = "Imputation number")
```



```r
#It looks fine so we will proceed to extract the imputed data

chained_train = complete(data=chained_train)

sapply(chained_train, function(x) sum(is.na(x)))
```

```
##            pcos             age          weight          height             bmi
##               0               0               0               0               0
##     blood_group      pulse_rate              rr              hb           cycle
##               0               0               0               0               0
##    cycle_length marriage_status        pregnant no_of_abortions       i_betahcg
##               0               0               0               0               0
##      ii_betahcg             fsh              lh     fsh_lh_ratio             hip
##               0               0               0               0               0
##           waist waist_hip_ratio             tsh             amh             prl
##               0               0               0               0               0
##           vitd3             prg             rbs     weight_gain     hair_growth
##               0               0               0               0               0
##  skin_darkening       hair_loss         pimples       fast_food    reg_exercise
##               0               0               0               0               0
##     bp_systolic    bp_diastolic    follicle_no_l   follicle_no_r     avg_f_size_l
##               0               0               0               0               0
##    avg_f_size_r     endometrium
##               0               0
```

Define the models

For the aim of this project, we will develop 4 different models with an increasing number of variables according to its easiness to collect. Then, model 1 will contain only variables that are collected throught the patient history, model 2 will have clinical examination ones added, model 3 blood tests and model 4 all the relevant variables available in the dataset.

It is worth mentioning that we removed some non-relevant variables from the datset based on the scientific literature of the field. A more detailed information about this selection can be found in the final report available in this repository.

```
# Create dataset 1 using only variables obtained through through patient history
model1_vars = c("pcos","age","cycle","cycle_length",
                 "no_of_abortions", "weight_gain", "hair_growth", "skin_darkening",
                 "hair_loss","pimples","fast_food", "reg_exercise")

# Create dataset 2 using variables obtained through patient history and clinical examination
model2_vars  = c(model1_vars, "weight","height","bmi",
                 "hip", "waist","waist_hip_ratio",
                 "bp_systolic", "bp_diastolic")

# Create dataset 3 using variables obtained through patient history, clinical examination and blood tes
model3_vars = c(model2_vars, "fsh", "lh", "fsh_lh_ratio",
              "amh", "prl", "vitd3", "prg", "rbs")

# Create dataset 4 using variables obtained through patient history, clinical examination, blood tests
model4_vars = c(model3_vars, "follicle_no_l", "follicle_no_r" ,
              "avg_f_size_l", "avg_f_size_r", "endometrium")
```

Logistic regression modeling

```
fitControl <- trainControl(
    method = 'cv',
    number = 5,
    savePredictions = 'final',
    classProbs = TRUE,
    summaryFunction=twoClassSummary)

set.seed(504)
#Since we have few samples, I will use bootstrapping instead of cross fold validation

models_logreg = list()

for (model in list(model1_vars, model2_vars, model3_vars, model4_vars)){
  cv_model = caret::train(
    pcos ~ .,
    data = chained_train %>%
      select(all_of(model)),
    method = "glm",
    family = "binomial",
    trControl = fitControl)
  models_logreg = append(models_logreg, list(cv_model))
}
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.

## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.

## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
names(models_logreg) = c("model1","model2", "model3", "model4")

models_logreg
```

```
## $model1
## Generalized Linear Model
##
## 379 samples
##  11 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 304, 303, 303, 303, 303
## Resampling results:
##
##   ROC        Sens       Spec
##   0.9050523  0.9019608  0.718
##
##
## $model2
## Generalized Linear Model
##
## 379 samples
##  19 predictor
##   2 classes: 'No', 'Yes'
##
```

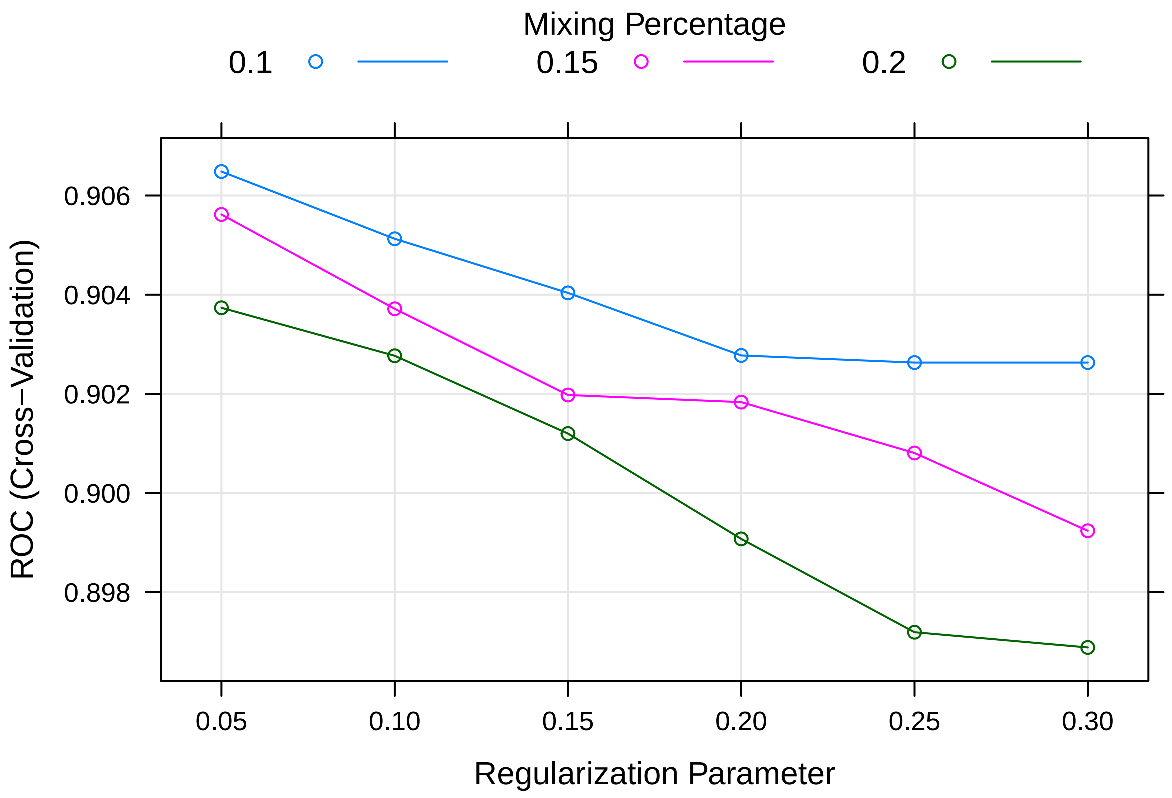
```
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 303, 304, 303, 303, 303
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8943791  0.9019608  0.6946667
##
##
## $model3
## Generalized Linear Model
##
## 379 samples
##  27 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 303, 304, 303, 303, 303
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8718627  0.8941176  0.6943333
##
##
## $model4
## Generalized Linear Model
##
## 379 samples
##  32 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 303, 304, 303, 303, 303
## Resampling results:
##
##   ROC        Sens       Spec
##   0.9350327  0.9137255  0.815
```

Elastic net regression

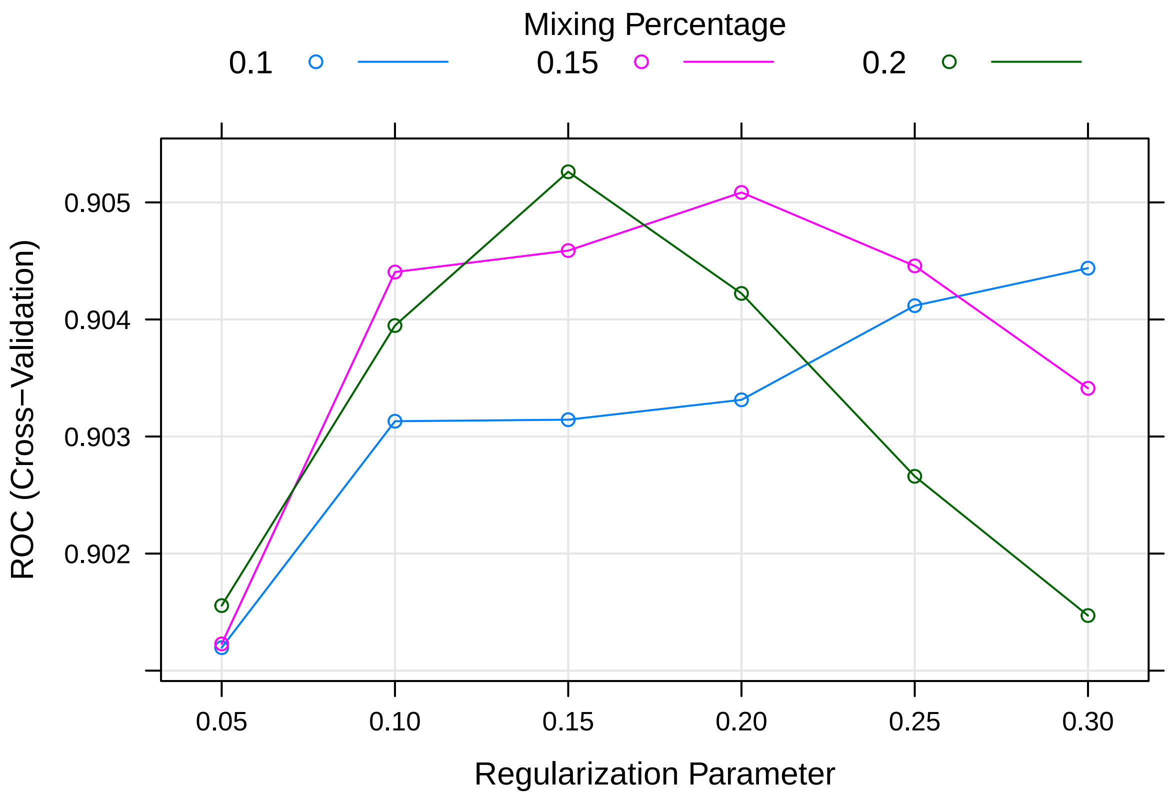
```
models_EN = list()

for (model in list(model1_vars, model2_vars, model3_vars, model4_vars)){
  en_model = caret::train(
    pcos ~ .,
    trControl = fitControl,
    data = chained_train %>%
      select(all_of(model)),
    method = "glmnet",
    tuneGrid = expand.grid(alpha = seq(0.1,.2,by = 0.05),
                           lambda = seq(0.05,0.3,by = 0.05)),
    verbose = FALSE,
    metric="ROC")
  models_EN = append(models_EN, list(en_model))
}

names(models_EN) = c("model1","model2", "model3", "model4")
models_EN
```

```
## $model1
## glmnet
##
## 379 samples
##  11 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 303, 304, 303, 303, 303
## Resampling results across tuning parameters:
##
##   alpha  lambda  ROC        Sens       Spec
##   0.10   0.05    0.9064837  0.9254902  0.7253333
##   0.10   0.10    0.9051275  0.9372549  0.6933333
##   0.10   0.15    0.9040359  0.9490196  0.6613333
##   0.10   0.20    0.9027745  0.9529412  0.6373333
##   0.10   0.25    0.9026307  0.9568627  0.6053333
##   0.10   0.30    0.9026307  0.9568627  0.5650000
##   0.15   0.05    0.9056176  0.9254902  0.7253333
##   0.15   0.10    0.9037157  0.9411765  0.6933333
##   0.15   0.15    0.9019771  0.9490196  0.6533333
##   0.15   0.20    0.9018333  0.9568627  0.6133333
##   0.15   0.25    0.9008072  0.9568627  0.5730000
##   0.15   0.30    0.8992386  0.9568627  0.5086667
##   0.20   0.05    0.9037353  0.9215686  0.7253333
##   0.20   0.10    0.9027680  0.9411765  0.6773333
##   0.20   0.15    0.9011993  0.9529412  0.6533333
##   0.20   0.20    0.8990752  0.9568627  0.5810000
##   0.20   0.25    0.8971928  0.9568627  0.5246667
##   0.20   0.30    0.8968856  0.9568627  0.4446667
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.1 and lambda = 0.05.
##
## $model2
## glmnet
##
## 379 samples
##  19 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 304, 303, 303, 303, 303
## Resampling results across tuning parameters:
##
##   alpha  lambda  ROC        Sens       Spec
##   0.10   0.05    0.9011961  0.9215686  0.6930000
##   0.10   0.10    0.9031307  0.9372549  0.6770000
##   0.10   0.15    0.9031438  0.9490196  0.6610000
##   0.10   0.20    0.9033137  0.9490196  0.6366667
##   0.10   0.25    0.9041176  0.9529412  0.5876667
##   0.10   0.30    0.9044379  0.9529412  0.5636667
##   0.15   0.05    0.9012288  0.9254902  0.6933333
##   0.15   0.10    0.9044052  0.9411765  0.6853333
##   0.15   0.15    0.9045882  0.9450980  0.6526667
##   0.15   0.20    0.9050850  0.9529412  0.5880000
##   0.15   0.25    0.9044575  0.9529412  0.5636667
##   0.15   0.30    0.9034118  0.9607843  0.5316667
##   0.20   0.05    0.9015556  0.9294118  0.7013333
##   0.20   0.10    0.9039477  0.9372549  0.6770000
##   0.20   0.15    0.9052614  0.9411765  0.6363333
##   0.20   0.20    0.9042222  0.9529412  0.5956667
##   0.20   0.25    0.9026601  0.9568627  0.5556667
##   0.20   0.30    0.9014706  0.9647059  0.4590000
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.2 and lambda = 0.15.
##
## $model3
## glmnet
##
## 379 samples
##  27 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 304, 303, 303, 303, 303
## Resampling results across tuning parameters:
##
##   alpha  lambda  ROC        Sens       Spec
##   0.10   0.05    0.8999869  0.9058824  0.6930000
##   0.10   0.10    0.9050327  0.9176471  0.6930000
##   0.10   0.15    0.9045882  0.9294118  0.6686667
##   0.10   0.20    0.9049150  0.9372549  0.6366667
##   0.10   0.25    0.9041111  0.9450980  0.6206667
##   0.10   0.30    0.9040980  0.9529412  0.5806667
##   0.15   0.05    0.9015686  0.9058824  0.6930000
##   0.15   0.10    0.9049085  0.9176471  0.6930000
##   0.15   0.15    0.9038105  0.9294118  0.6526667
##   0.15   0.20    0.9033203  0.9411765  0.6286667
##   0.15   0.25    0.9032941  0.9529412  0.5806667
##   0.15   0.30    0.9017190  0.9529412  0.5323333
##   0.20   0.05    0.9031699  0.9098039  0.6930000
##   0.20   0.10    0.9046144  0.9215686  0.6930000
##   0.20   0.15    0.9025294  0.9333333  0.6366667
##   0.20   0.20    0.9025359  0.9529412  0.6126667
##   0.20   0.25    0.9009542  0.9529412  0.5483333
##   0.20   0.30    0.9011111  0.9568627  0.4603333
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.1 and lambda = 0.1.
##
## $model4
## glmnet
##
## 379 samples
##  32 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 304, 303, 303, 303, 303
## Resampling results across tuning parameters:
##
##   alpha  lambda  ROC        Sens       Spec
##   0.10   0.05    0.9597255  0.9529412  0.7803333
##   0.10   0.10    0.9595621  0.9607843  0.7403333
##   0.10   0.15    0.9592288  0.9647059  0.7163333
##   0.10   0.20    0.9587451  0.9647059  0.7083333
##   0.10   0.25    0.9590784  0.9725490  0.7000000
##   0.10   0.30    0.9590784  0.9803922  0.6920000
##   0.15   0.05    0.9597320  0.9490196  0.7723333
##   0.15   0.10    0.9597190  0.9607843  0.7323333
##   0.15   0.15    0.9597124  0.9647059  0.7163333
##   0.15   0.20    0.9582745  0.9686275  0.7000000
##   0.15   0.25    0.9573203  0.9843137  0.6920000
##   0.15   0.30    0.9573268  0.9843137  0.6513333
##   0.20   0.05    0.9587647  0.9490196  0.7723333
##   0.20   0.10    0.9595556  0.9568627  0.7403333
##   0.20   0.15    0.9585948  0.9647059  0.7163333
##   0.20   0.20    0.9579673  0.9764706  0.7000000
##   0.20   0.25    0.9582876  0.9843137  0.6430000
##   0.20   0.30    0.9594052  0.9882353  0.6270000
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.15 and lambda = 0.05.
```

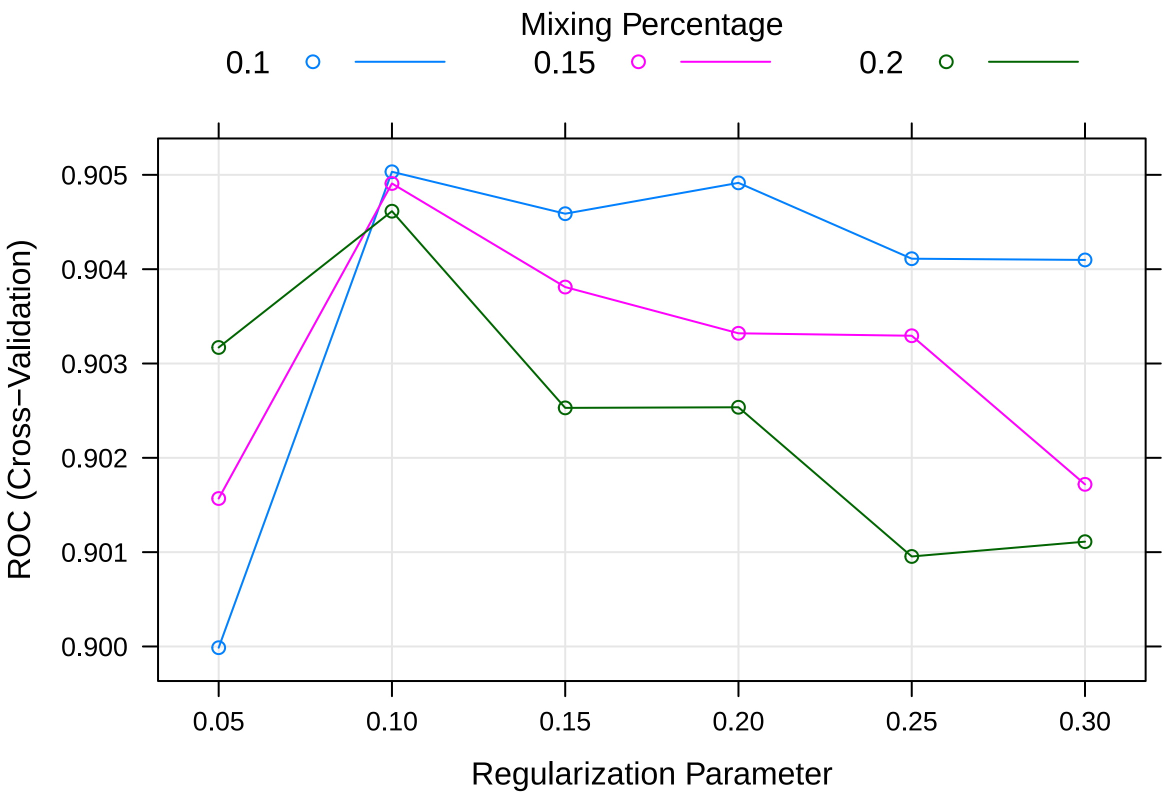
```
#Plot parameter tuning of model 1
plot(models_EN[[1]])
```



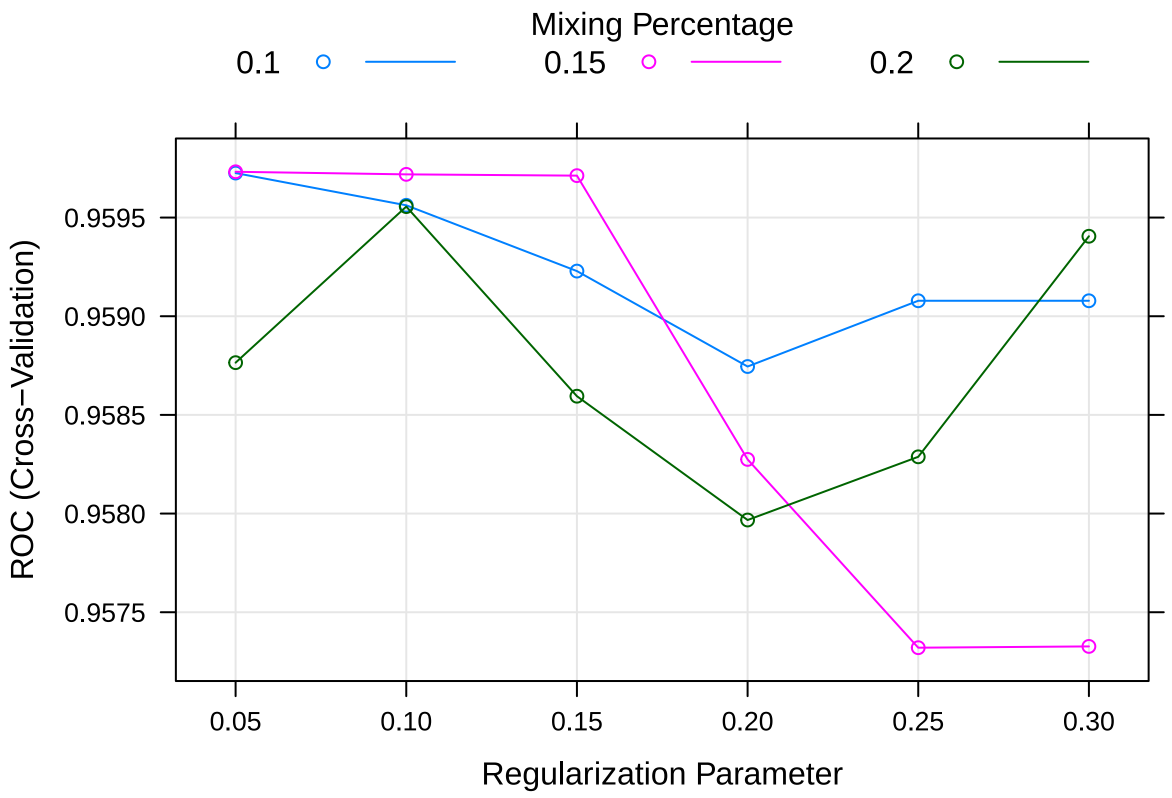
```
#Plot parameter tuning of model 2
plot(models_EN[[2]])
```



```
#Plot parameter tuning of model 3
plot(models_EN[[3]])
```



```r
#Plot parameter tuning of model 4
plot(models_EN[[4]])
```



Random Forest

Model comparison

Predict on the validation data set and get statistics in a table

*Select the best one at the end

Effect of class imbalance on the best model (if time allows)