Heart Failure analysis and modeling - code and outputs

Alexander Kheirallah

07/10/2020

This HTML/PDF file is accompanied by explanatory `README` as well as `report.pdf` both hosted on this repository. Below you can find the *code* and *comment* sections. In the *comment* section I elaborate in more detail about the analysis and choice of specific approaches.

Code section

Data processing

Data formatting and value modifications

```
d = read.table(here("data","processed.cleveland.data"),sep = ",",stringsAsFactors = F)
features = c("age","sex","cp","trestbps","chol","fbs","restecg","thalach","exang","oldpeak","slope","ca
names(d) = c(features,"Y")
d$ca = as.numeric(d$ca)
d$thal = as.numeric(d$thal)
d$slope[d$slope==3]=1
d$slope[d$slope==1]=3
d$thal[d$thal==3]=1
d$thal[d$thal==7]=2
d$thal[d$thal==6]=3
d$cp = as.factor(d$cp)
d$restecg = as.factor(d$restecg)
```

Checking what variables have missing data

```
colSums(is.na(d))
```

```
##      age      sex       cp trestbps     chol      fbs  restecg  thalach
##        0        0        0        0        0        0        0        0
##    exang  oldpeak    slope       ca     thal        Y
##        0        0        0        4        2        0
```

Removing rows with at least one NA and checking the size of resulting table

```
keep = !(is.na(d$ca) | is.na(d$thal))
d = d[keep,]
dim(d)
```

```
## [1] 297  14
```

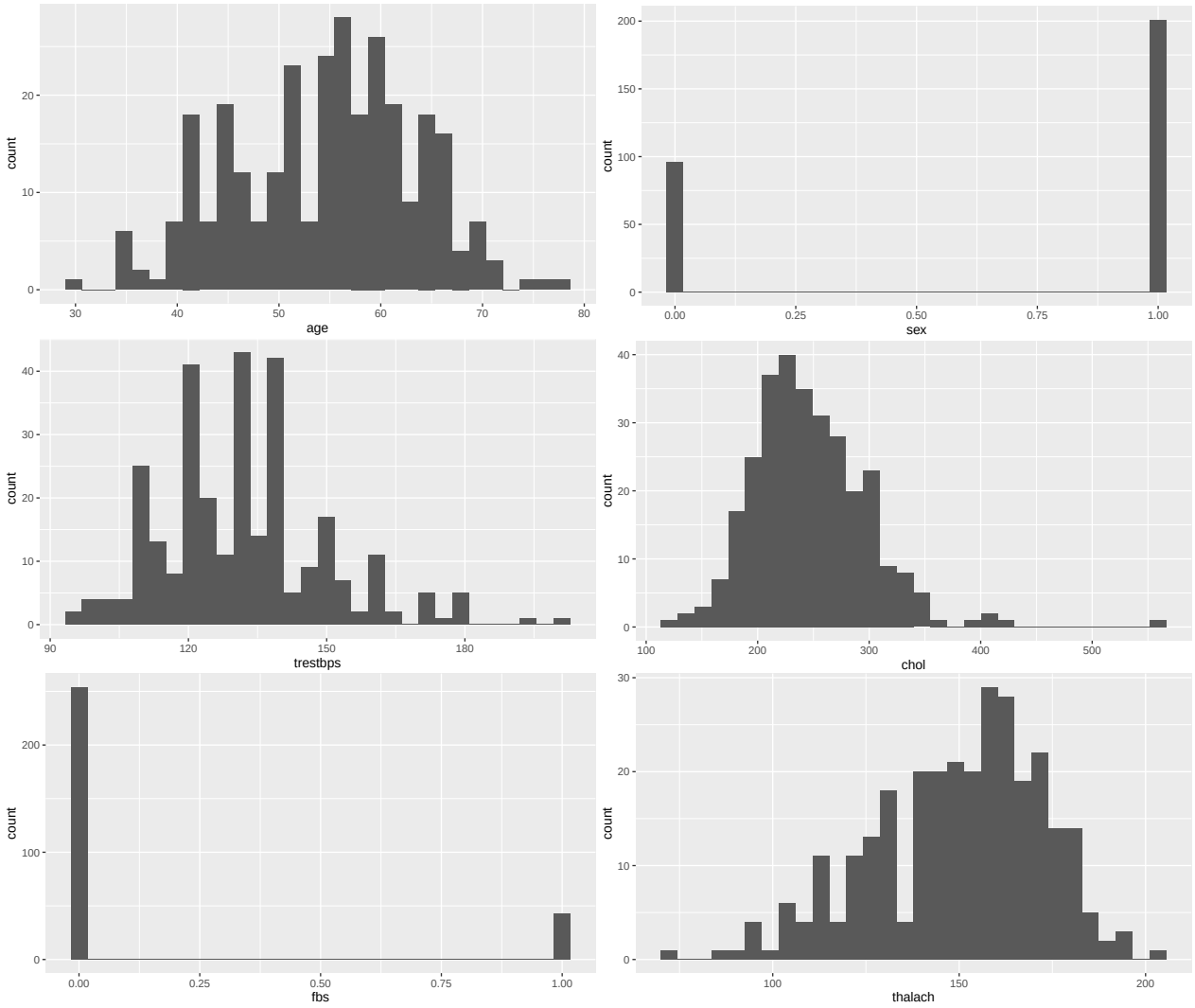
Converting outcome to a binary label and checking relative proportions of diseased and healthy individuals

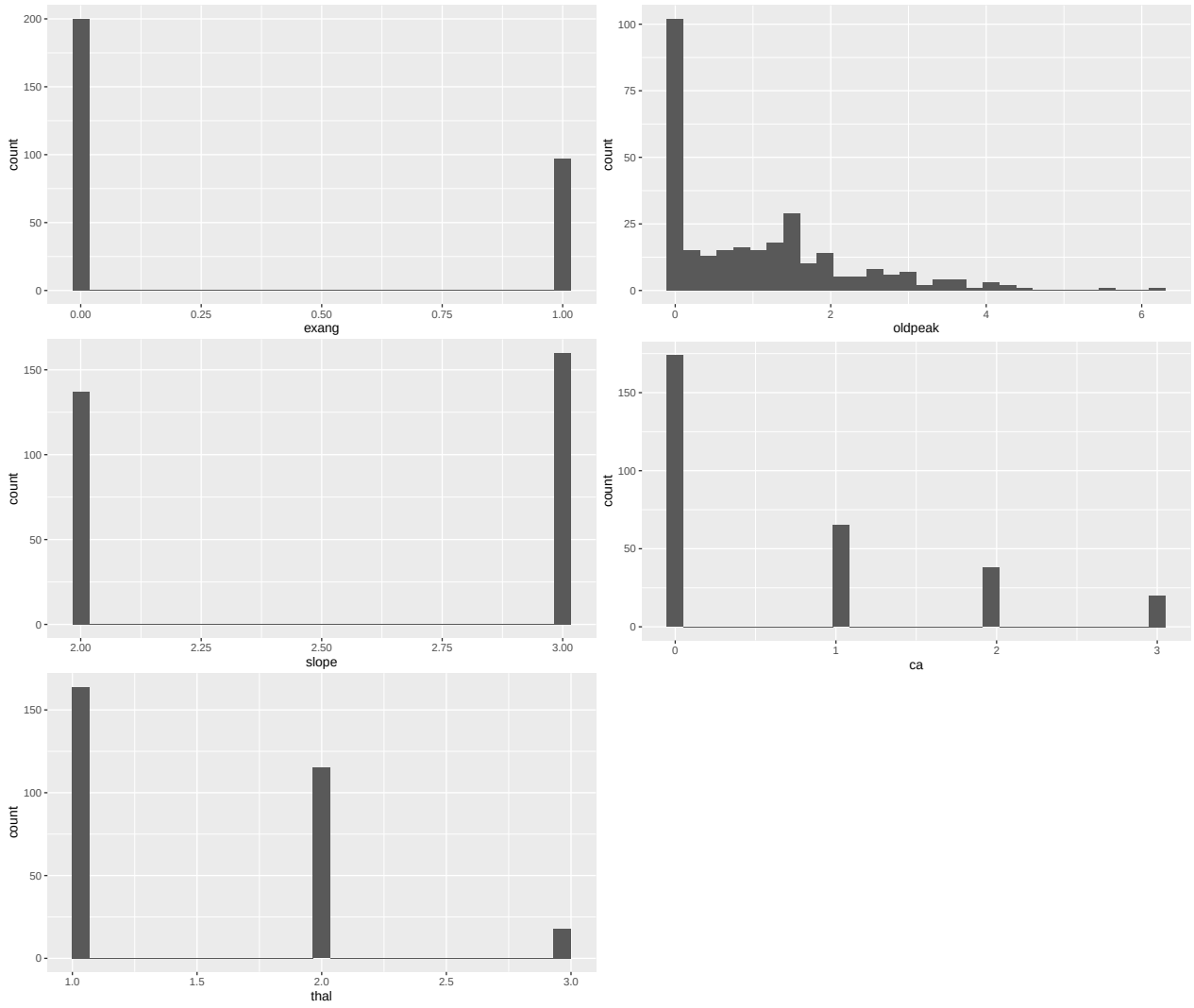
```
d = d %>%
  mutate(Y=Y>0)
table(d$Y)
```

```
##
## FALSE  TRUE
##   160   137
```

Plotting data distributions

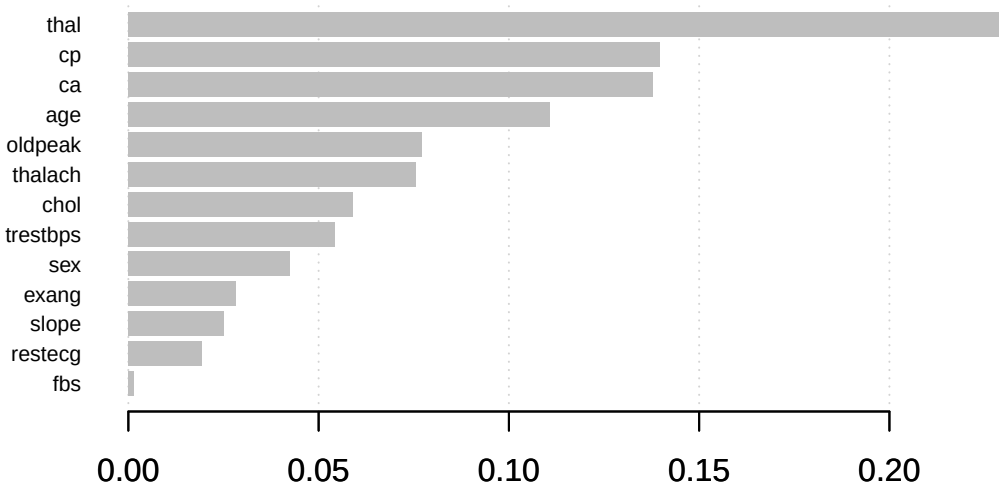
```
for(f in features[-match(c("cp","restecg"),features)]){
  print(d %>%
    ggplot(aes_string(x = f)) +geom_histogram())
}
```





Calculating feature importance

```
X = d[,features]
Y = d[,"Y"]
parameters = list(booster = "gbtree",
                  objective = "binary:logistic")

xgb = xgboost(data = data.matrix(X),
              label = Y,
              nrounds = 1000,
              verbose = F,
              params = parameters)

importance_matrix = xgb.importance(model = xgb,feature_names = features)
xgb.plot.importance(importance_matrix[1:nrow(importance_matrix),])
```

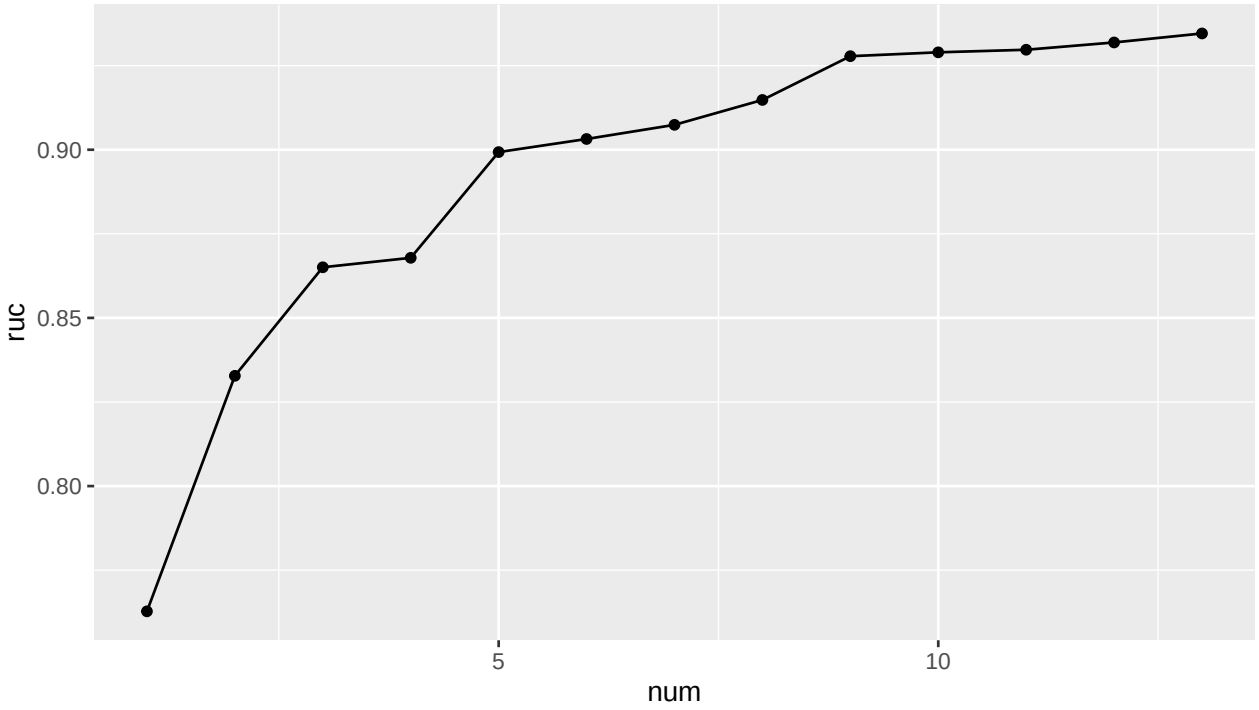


Undertaking the iterative model evaluation procedure with plotting of AUC values for both train and test partitions

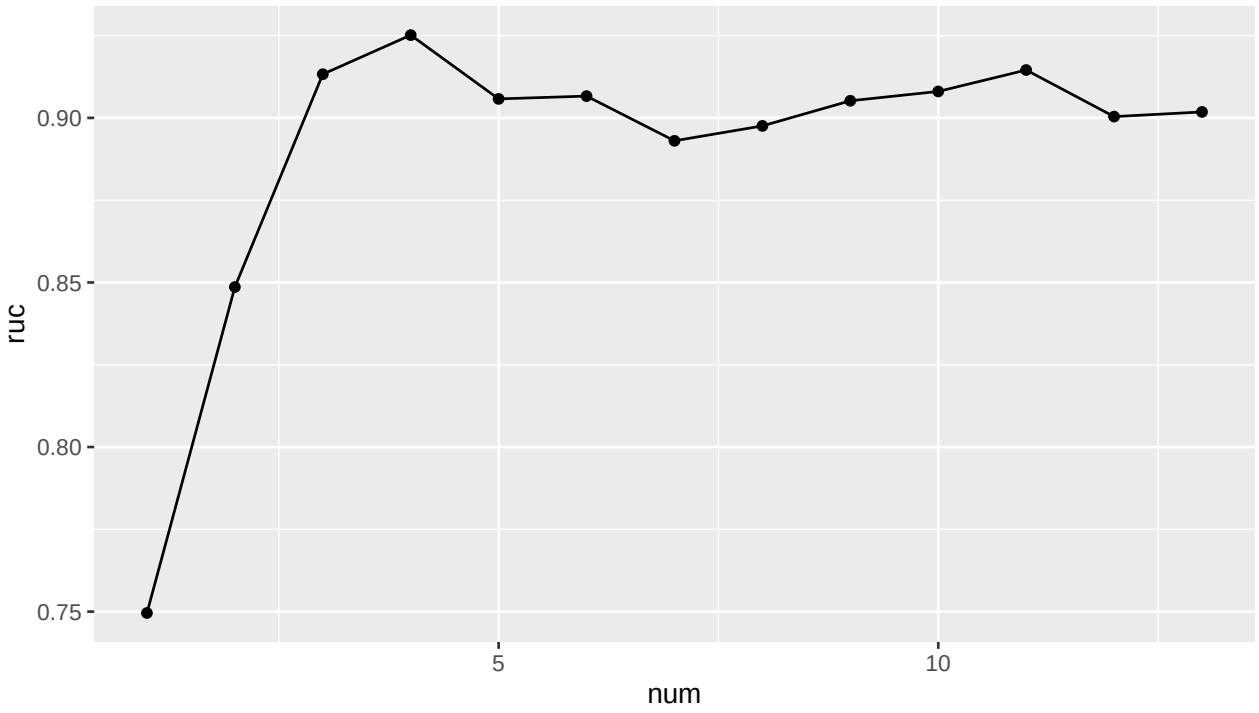
```
allFeatures_ordered = importance_matrix$Feature

splits = train_test_split(d = d,train_size = 0.6,seed = 123)
train = splits$train
test = splits$test

act_train = train$Y
act_test = test$Y

RUCs_train = c()
RUCs_test = c()

x = allFeatures_ordered[1]
form = paste("Y ~",x)
fit = glm(form, data = train, family = "binomial")

train_pred = predict(fit,newdata = train,type = "response")
roc_train = roc(response = act_train,predictor = train_pred,plot=F)
RUCs_train = c(RUCs_train,roc_train$auc)

test_pred = predict(fit,newdata = test,type = "response")
roc_test = roc(response = act_test,predictor = test_pred,plot=F)
RUCs_test = c(RUCs_test,roc_test$auc)

for (i in 2:length(allFeatures_ordered)) {
  f = allFeatures_ordered[i]
  # update feature space
  x = paste(x,"+",f)
  form = paste("Y ~",x)
  fit = glm(form, data = train, family = "binomial")

  train_pred = predict(fit,newdata = train,type = "response")
  roc_train = roc(response = act_train,predictor = train_pred,plot=F)
```

```
  RUCs_train = c(RUCs_train,roc_train$auc)

  test_pred = predict(fit,newdata = test,type = "response")
  roc_test = roc(response = act_test,predictor = test_pred,plot=F)
  RUCs_test = c(RUCs_test,roc_test$auc)
}
```

```
RUCs_train_d = as.data.frame(cbind(num=1:length(RUCs_train),ruc=RUCs_train))
ggplot(RUCs_train_d, aes(x=num, y=ruc)) + geom_point() + geom_line()
```




```
RUCs_test_d = as.data.frame(cbind(num=1:length(RUCs_test),ruc=RUCs_test))
ggplot(RUCs_test_d, aes(x=num, y=ruc)) + geom_point() + geom_line()
```



Plotting the ROC curve with 4 most important features included

With seed 123 (same as in model evaluation procedure)

```
fit = glm(Y ~ thal + cp + ca + age, data = train, family = "binomial")
train_pred = predict(fit,newdata = train,type = "response")
test_pred = predict(fit,newdata = test,type = "response")

par(pty="s")
# train
roc(response = act_train,predictor = train_pred,
    plot=T, legacy.axes=T, percent=T, xlab = "False Positive %", ylab = "True Positive %", col="blue",p
```

```
## Setting levels: control = FALSE, case = TRUE
```

```
## Setting direction: controls < cases
```

```
##
## Call:
## roc.default(response = act_train, predictor = train_pred, percent = T,     plot = T, legacy.axes = T
##
## Data: train_pred in 98 controls (act_train FALSE) < 80 cases (act_train TRUE).
## Area under the curve: 86.79%
```

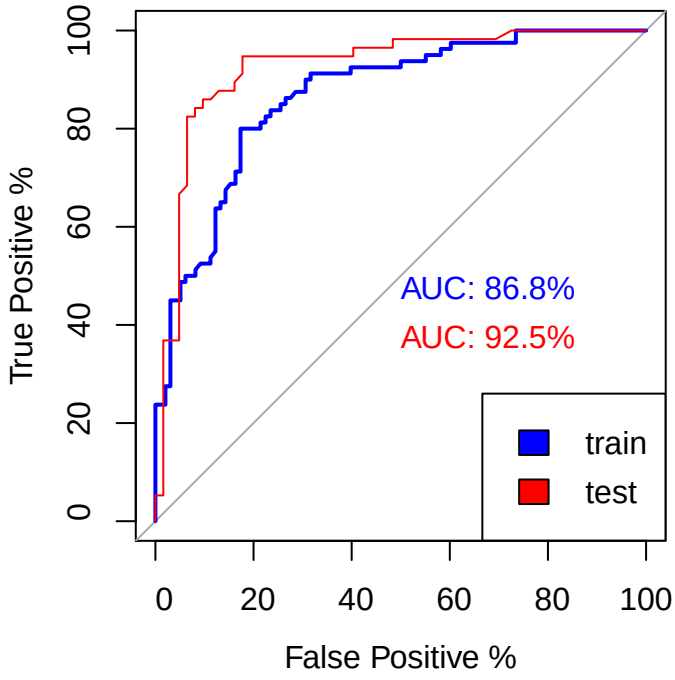
```
# test
roc(response = act_test,predictor = test_pred,
    percent=T, col="red", lwd=1, print.auc=T, add=T, print.auc.y=40, plot = T)
```

```
## Setting levels: control = FALSE, case = TRUE
## Setting direction: controls < cases
```

```
##
```

```
## Call:
## roc.default(response = act_test, predictor = test_pred, percent = T,     plot = T, col = "red", lwd =
##
## Data: test_pred in 62 controls (act_test FALSE) < 57 cases (act_test TRUE).
## Area under the curve: 92.52%
```

```
legend("bottomright",c("train","test"),fill=c("blue","red"))
```



With a new seed to test what happens with test's AUC as a result of this change

```
splits = train_test_split(d = d,train_size = 0.6,seed = 567)
train = splits$train
test = splits$test

act_train = train$Y
act_test = test$Y

fit = glm(Y ~ thal + cp + ca + age, data = train, family = "binomial")
train_pred = predict(fit,newdata = train,type = "response")
test_pred = predict(fit,newdata = test,type = "response")

par(pty="s")
# train
roc(response = act_train,predictor = train_pred,
    plot=T, legacy.axes=T, percent=T, xlab = "False Positive %", ylab = "True Positive %", col="blue",p
```

```
## Setting levels: control = FALSE, case = TRUE
```

```
## Setting direction: controls < cases
```

```
##
## Call:
```

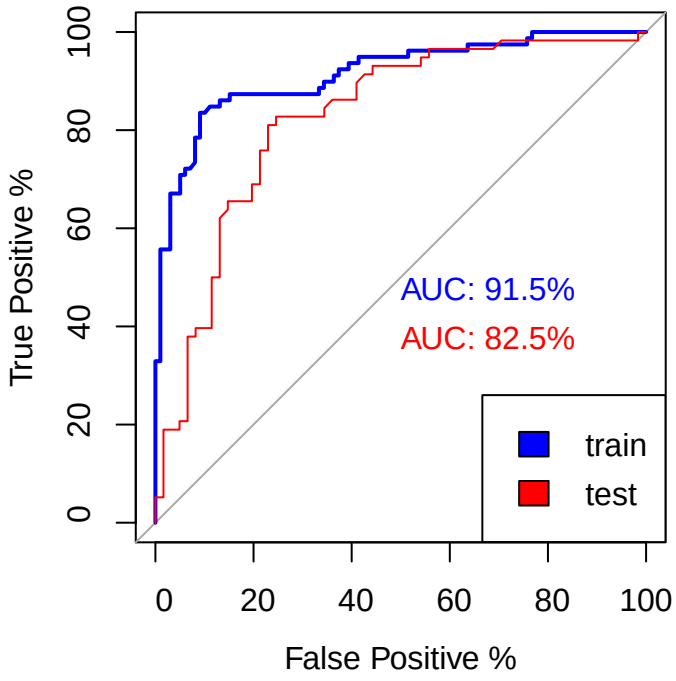
```
## roc.default(response = act_train, predictor = train_pred, percent = T,     plot = T, legacy.axes = T
##
## Data: train_pred in 99 controls (act_train FALSE) < 79 cases (act_train TRUE).
## Area under the curve: 91.5%
```

```
# test
roc(response = act_test,predictor = test_pred,
    percent=T, col="red", lwd=1, print.auc=T, add=T, print.auc.y=40, plot = T)
```

```
## Setting levels: control = FALSE, case = TRUE
## Setting direction: controls < cases
```

```
##
## Call:
## roc.default(response = act_test, predictor = test_pred, percent = T,     plot = T, col = "red", lwd =
##
## Data: test_pred in 61 controls (act_test FALSE) < 58 cases (act_test TRUE).
## Area under the curve: 82.5%
```

```
legend("bottomright",c("train","test"),fill=c("blue","red"))
```



Underataking statistical analysis of logistic regression coefficients to identify features correlated with desease status

```
##
## Call:
## glm(formula = Y ~ ., family = "binomial", data = d)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5888  -0.5236  -0.1560   0.3779   2.6623
##
## Coefficients:
```

```
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.096245   2.790348  -1.468 0.142103
## age         -0.011691   0.024119  -0.485 0.627873
## sex          1.672687   0.507174   3.298 0.000974 ***
## cp2          1.045189   0.741420   1.410 0.158625
## cp3          0.239435   0.642898   0.372 0.709572
## cp4          2.004446   0.636971   3.147 0.001650 **
## trestbps     0.024603   0.011075   2.221 0.026319 *
## chol         0.005799   0.003920   1.479 0.139025
## fbs         -0.777065   0.576378  -1.348 0.177599
## restecg1     0.853254   2.318662   0.368 0.712878
## restecg2     0.438757   0.372842   1.177 0.239279
## thalach     -0.016480   0.010540  -1.564 0.117906
## exang        0.817072   0.430710   1.897 0.057823 .
## oldpeak      0.390926   0.195877   1.996 0.045959 *
## slope       -1.014465   0.415549  -2.441 0.014636 *
## ca           1.270343   0.263438   4.822 1.42e-06 ***
## thal         0.701018   0.339845   2.063 0.039135 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 409.95  on 296  degrees of freedom
## Residual deviance: 200.00  on 280  degrees of freedom
## AIC: 234
##
## Number of Fisher Scoring iterations: 6
```

Comment section

Comment on data and its initial modification

Supplied data consists of 303-by-14 table. Luckily only 4 and 2 fields are missing `NA`s in `ca` and `thal` respectivelly. It is not a lot, so removing rows with at least one `NA` is OK as we're not getting rid of much data.

The outcome label that will be modelled is in column 14th and consists of discrete integers between `0` and `4`, presumably indicating the severity of angiographic disease vessels narrowing. However the task will be simplified by treating and value greater than `0` as `1` yielding as a binary classification challenge. Luckily we're not having to deal with severe class imbalance problem as 46% of data has a positive disease status (i.e. roughly equal amounts of diseased and healthy individuals).

Features consist of combination of binary, categorical and ordinal and continous numeric variables. Categorical features ought to be treated with caution as they are valued by whole integers and hence can be confused to have inherent numerical ordering. There are two categorical variables, `cp` and `restecg`. It will be important, especially from statistical learning point of view, to encode them as R environment factors which has been done. Moreover, it is advisable to slightly modify the ordinal features `slope` and `thal` (for the former by exchanging `1` by `3` and vice versa, while for latter by by exchanging `3` for `0`, `6` for `2` and `7` for `1`). This is because for `slope`, value `1` represents *up* and `3` represents *down* and it makes more sense to have the feature in ascending order. For `thal` numbers represent "fixation level" of defect and hence have inherent ordering but value `7` represents "reversiable defect", while `6` "fixed defect" and so mathematical ordering isn't preserved.

Comment on further data pre-processing

With regards to further data pre-processing, such as centering or standardisation, I've decided not to do any further processing but treat the data as it is. This is because my preference is to do the least amount of data modification as possiable and also because any centering or standardisation is not going to be of much added benefit for either random forest or logistic regression algorithms (which are explored). Although centering values may aid in the interpretation of regression coefficients as then they mean expected change in outcome for unit increase in feature while the rest of feature are average instead of zero (see **here** for discussion of pros and cons of centering and standardisation), I don't feel like this is necessery. Having said that this **paper** suggests that random forest derived feature importance may be misleading because of a biased variable selection for continous numerical variables relative to discerete or categorical variables given "bigger space" of decision boundery points for continous numerical variables in decision tree. As we're dealing with mix of feature types this may be an issue but it can be addressed, according to the authors of the paper, by avoiding replacment when sampling during bagging procedure. Lucklily `XGboost` package that I will use for feature importance implements bagging without replacement (see **reference** under caveats). Finally numerical variables appear normally distributed with no evidence for skew no warrant a transformation.