MACHINE LEARNING PROJECT

Student : Pop Diana-Stefania

Project Plan - Detailed Description

The purpose of the project is to determine whether a person has a heart disease based on multiple factors.

Attribute Information:

-- Only 14 used

-- 1. #3 (age)

-- 2. #4 (sex) (1 = male; 0 = female)

-- 3. #9 (cp) chest pain type

-- Value 1: typical angina

-- Value 2: atypical angina

-- Value 3: non-anginal pain

-- Value 4: asymptomatic

-- 4. #10 (trestbps) resting blood pressure

-- 5. #12 (chol) serum cholesterol in mg/dl

-- 6. #16 (fbs) (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

-- 7. #19 (restecg) resting electrocardiographic results

-- 8. #32 (thalach) maximum heart rate achieved

-- 9. #38 (exang) exercise induced angina (1 = yes; 0 = no)

-- 10. #40 (oldpeak) ST depression induced by exercise relative to rest

-- 11. #41 (slope) the slope of the peak exercise ST segment

-- Value 1: upsloping

-- Value 2: flat

-- Value 3: downsloping

-- 12. #44 (ca) number of major vessels (0-3)

-- 13. #51 (thal) 3 = normal; 6 = fixed defect; 7 = reversable defect

-- 14. #58 (num) (the predicted attribute)

-- Complete attribute documentation: see heart-disease.names

Data Set Information:

This directory contains 4 databases concerning heart disease diagnosis.
All attributes are numeric-valued. The data was collected from the
four following locations:

1. Cleveland Clinic Foundation (cleveland.data)
2. Hungarian Institute of Cardiology, Budapest (hungarian.data)
3. V.A. Medical Center, Long Beach, CA (long-beach-va.data)
4. University Hospital, Zurich, Switzerland (switzerland.data)

Each database has the same instance format. While the databases have 76
raw attributes, only 14 of them are actually used. Thus I've taken the
liberty of making 2 copies of each database: one with all the attributes
and 1 with the 14 attributes actually used in past experiments.

I. Prepare problem

- *Load libraries*

```
library(mlbench)
library(e1071)
library(lattice)
library(corrplot)
library(caret)
...
```

- *Load dataset*

```
data<-read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases
/heart-disease/processed.cleveland.data")
```

- *Split into train/test data*

```
train_index <- sample(x=1:nrow(data), size=0.8*nrow(data))
train = data[train_index,]
test = data[-train_index,]
```

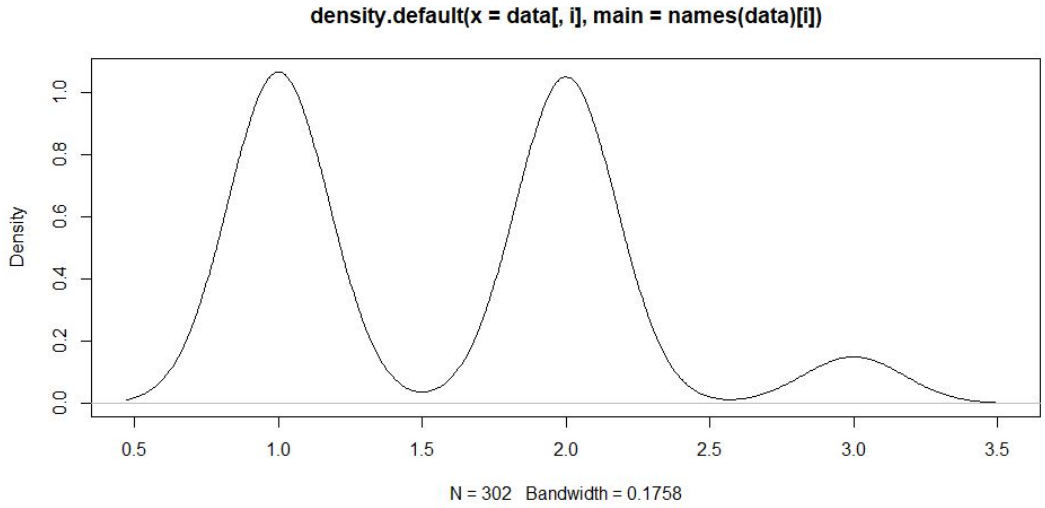
II. Summarize Data

- *Descriptive statistics*

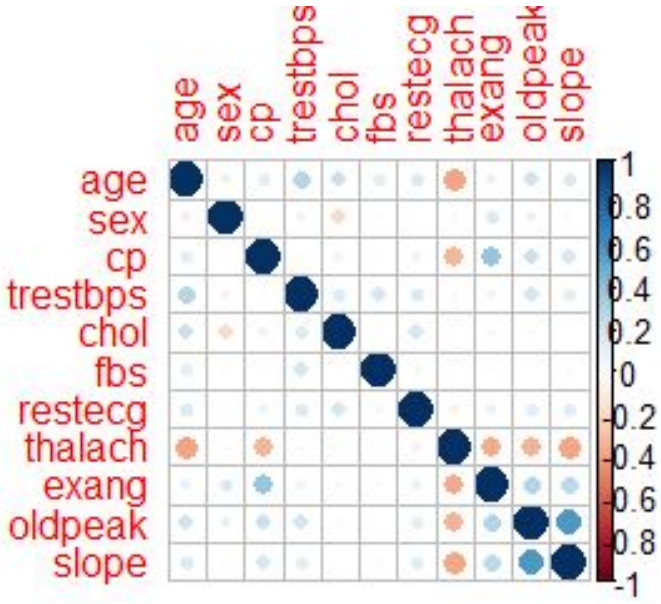
```
names(data) <- c( "age", "sex", "cp", "trestbps", "chol","fbs", "restecg",
"thalach","exang", "oldpeak","slope", "ca", "thal", "num")

# peek at data
head(data)

> dim(data)
[1] 302  14

# standard deviation
sapply(data[,1:11],sd)

 age       sex        cp   trestbps      chol       fbs
 9.0226986  0.4693508  0.9563024 17.7185161 51.9956455  0.3519800
   restecg    thalach      exang    oldpeak      slope
 0.9949140 22.9935210  0.4705889  1.1635071  0.6132045

# skewness
skew <- apply(data[,1:11],2,skewness)
print(skew)

age        sex         cp    trestbps       chol        fbs
-0.20769351 -0.74107083 -0.84462943  0.70611703  1.11227190
2.01438195
    restecg     thalach       exang     oldpeak       slope
 0.01335469 -0.52465125  0.72490720  1.25243785  0.50022497

# classification by sex
>data$sex<-ifelse(data$sex> 0,"male","female")
>table(data$sex)
female   male
   97    201

# classification by sex and disease
>sex_disease<-table(gender=data$sex,disease=data$num)
gender   disease no disease
  female      25         72
  male       113         88
```

- *Data visualization*

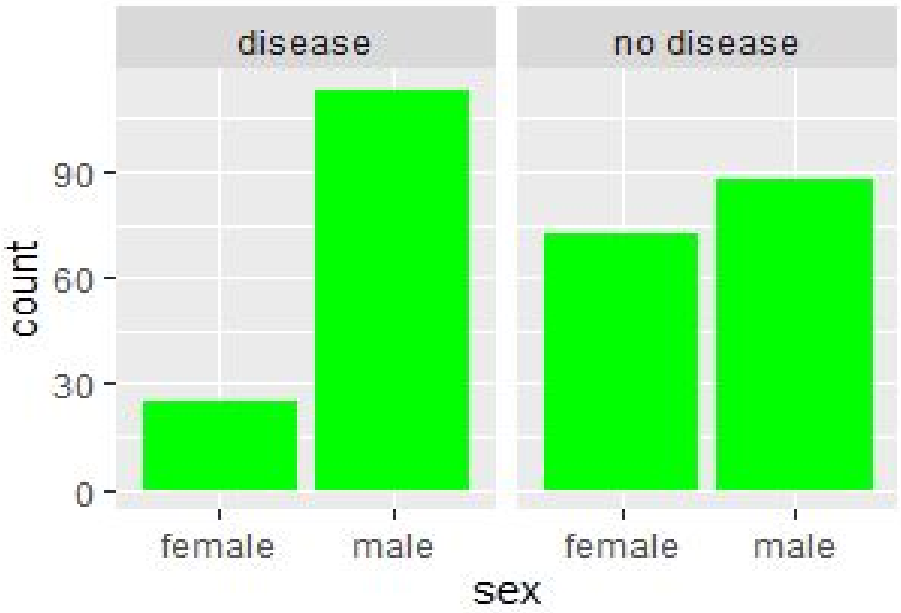
```
# density plot
for(i in 1:11)
   plot(density(data[,i], main=names(data)[i]))
```



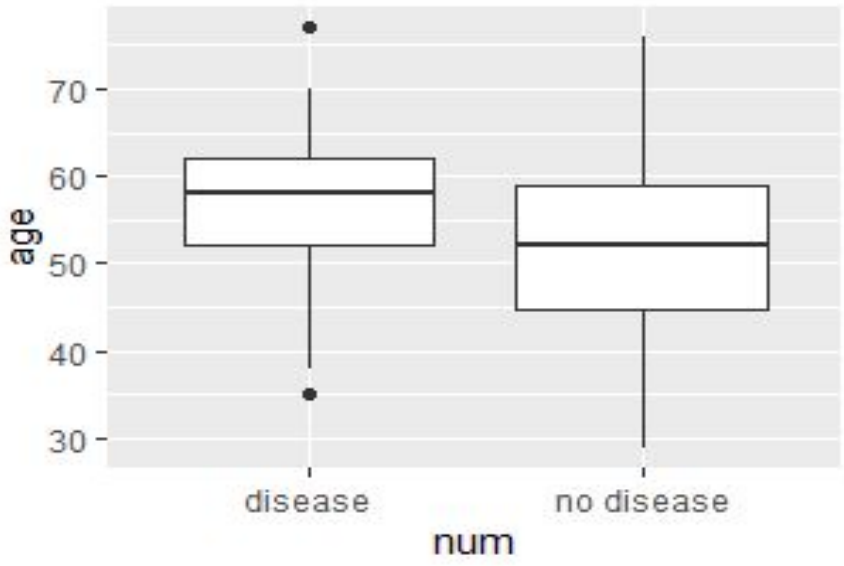
```
# correlation plot
corrplot(correlations, method="circle")
```




```
ggplot(data, aes(x=sex)) + geom_bar(fill="green") + facet_wrap(~num)
```



```
ggplot(data, aes(x=num,y=age)) + geom_boxplot()
```



III. Prepare Data

- *Data cleaning and transforms*

```
# find if any data is unavailable
> anyNA(data)
[1] FALSE

# eliminate rows with not found info
> which(data$ca %in% c("?"))
[1] 166 192 287 302
> data <- data[-c(166, 192, 287,302), ]
```

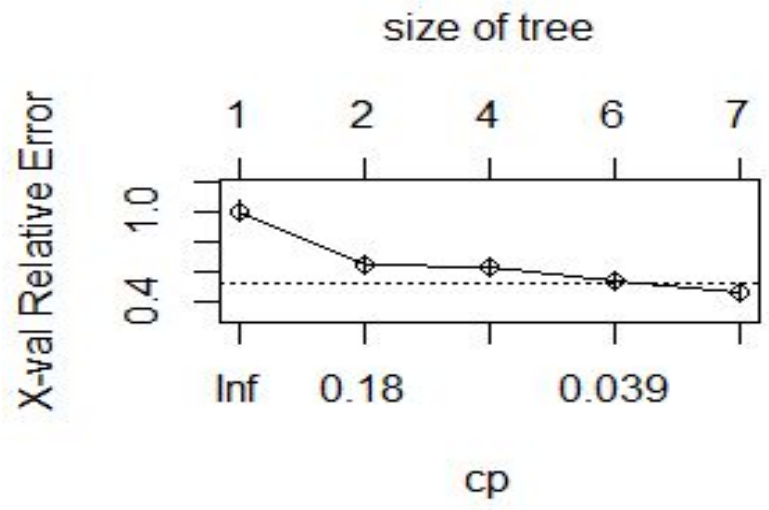
IV. Evaluate algorithms

1. Binomial linear model

```
fit<-glm(num~.,data=train, family="binomial")
pred_binomial<-predict(fit,newdata = test,type = "response")

# choose model in a stepwise algorithm
>stepAIC(fit, direction = "backward")
…
Call : glm(formula = num ~ sex + cp + trestbps + fbs + restecg + thalach +
    oldpeak + slope + ca + thal, family = "binomial", data = train)

fit_formula<-glm(formula = num ~ sex + cp + trestbps + restecg + thalach
+ slope +  exang + oldpeak + slope + ca + thal, family = "binomial",
         data = train)

pred_formula<-predict(fit_formula,newdata = test,type = "response")
```

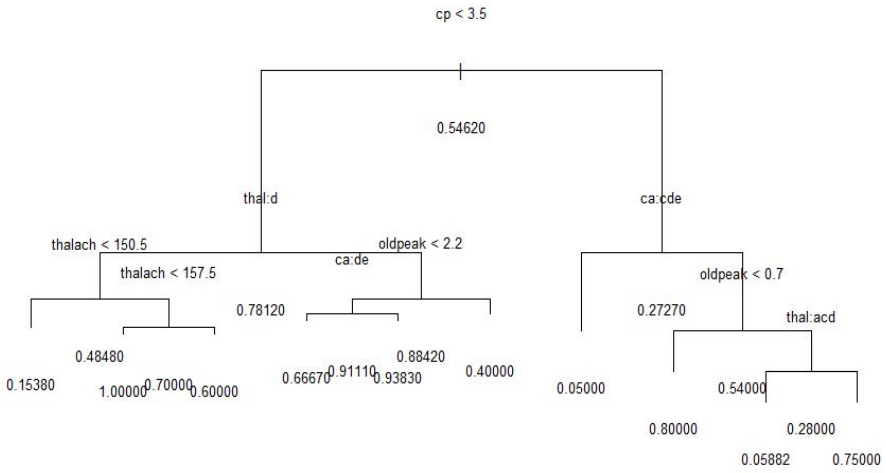
2. *Classification tree*

```
# recursive partitioning and classification tree model
fit_rpart <- rpart(formula = num ~ sex + cp + trestbps + restecg + thalach +
exang + oldpeak + slope + ca + thal, method = "class",
data = train)

plotcp(fit_rpart)
```




```
set.seed(123)

fit_rpart <- tree(num ~ sex + cp + trestbps + restecg + thalach +
exang + oldpeak + slope + ca + thal, data=train)
```

```
plot(fit_rpart, uniform=TRUE, main="Heart Attack Prediction")
text(fit_rpart, use.n=TRUE, all=TRUE, cex=.8)
```



3. Support vector machine

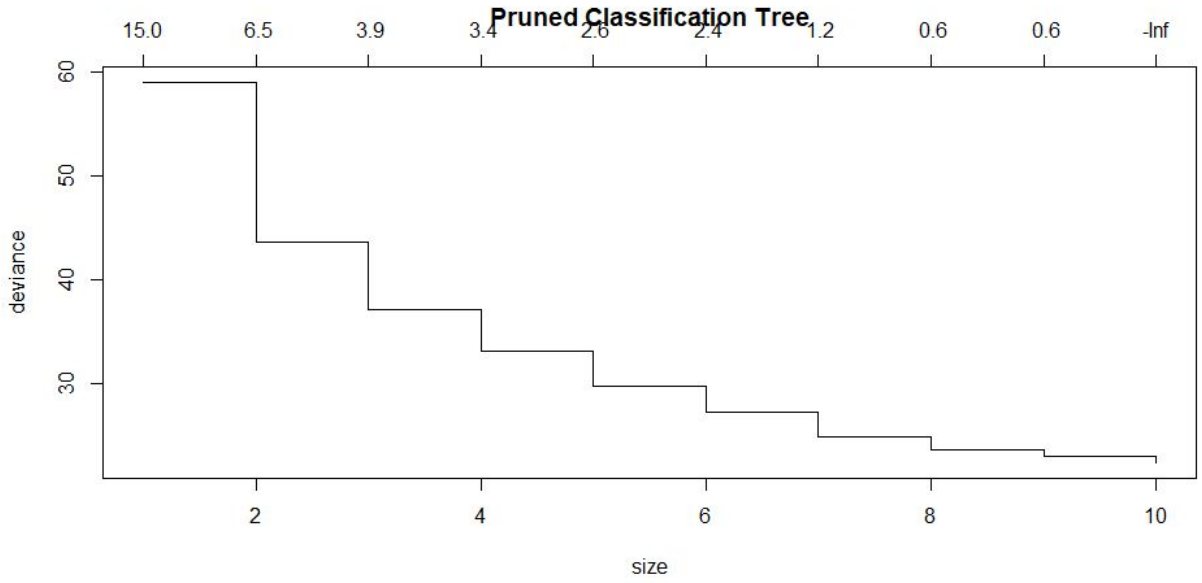
```
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3,
classProbs = TRUE)

svm_model = svm(num ~., data = train, method = "svmLinear",
                trControl=trctrl,
                preProcess = c("center", "scale"),
                tuneLength = 10)

svm_pred = predict(svm_model,test, type="vector")
```

V. Improve Accuracy

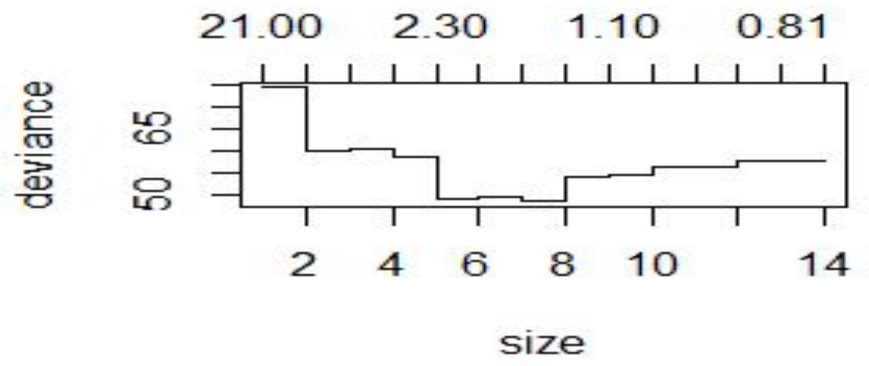
Prune the tree:

```
fit_rpart_pruned<- prune(fit_rpart)
```

```
plot(fit_rpart_pruned, uniform=TRUE, main="Pruned Classification Tree")
```




```
# try to improve test fit
pruned_tree <- prune.tree(fit_rpart, best=6)

pruned_prediction <- predict(pruned_tree, test, type="vector")

# use cross validation to find the best tree
tree_model <- tree(num ~ ., data=data)

cv_model <- cv.tree(tree_model)

plot(cv_model)
```




```
>best_size <- cv_model$size[which(cv_model$dev==min(cv_model$dev))]
>best_size
[1] 7

cv_model_pruned <- prune.rpart(tree_model,cp=7)
pruned_prediction <- predict(cv_model_pruned, test, type="vector")
```

VI. Save models for later use

```
saveRDS(fit, "./models/binomial.rds")
saveRDS(fit_formula, "./models/formula.rds")
saveRDS(cv_model_pruned, "./models/tree.rds")
saveRDS(svm_model, "./models/svm.rds")
```