CM3111 Big Data Coursework

Petar Bonchev

November 30, 2018

1 Data Exploration

1.1. Data choice

The dataset I decided to use is Phishing websites provided by UCI machine learning repository- https://archive.ics.uci.edu/ml/datasets/Phishing+Websites.

I chose this data set because of its practical potential to predict dangerouse websites for the user and also because the size of the dataset is more appropriate for the current coursework and of manageable size. In addition, hacking/ethical hacking have always been an interesting subject for me.

1.2. Problem statement and Data Exploration

In this dataset are presented one of the most famous and widely used features that have proved to be sound and effective in predicting phishing websites. Each attribute corresponds to a particular technique that was used to determine whether the tested website falls into one of the three website caterogies - legimate, suspicious and phishing.

In this coursework the aim is to build a predictive model so it can predict which websites are phishing websites and contain malware harmful to the user.
As we can see below the names of the features(columns), each column stands for a different way a website can be detected whether it contains some kind of malware. Although, this data set contains data for 31 distinctive malware detecting methods, there is no agreement in literature on the definitive features that characterize phishing websites and it is difficult to shape a dataset that covers all possible features. In addition, in this data set some new features have been proposed, new rules have been experimentally assigned to some well-known features and some other features have been updated.

-
```
options(scipen = 999) # disable scientific notation -numbers
cwFile <- read.csv('C:\\Users\\Peter Boncheff\\Desktop\\Courseworks\\Big Data\\phishing.csv')
# Read and save dataset
df <- cwFile # use an alternative data frame
cat("This phishing Websites database has", nrow(df), "rows and", ncol(df), "columns")

## This phishing Websites database has 999 rows and 31 columns

names(df) # illustrating all names of columns

##  [1] "having_IP_Address"           "URL_Length"
##  [3] "Shortining_Service"          "having_At_Symbol"
##  [5] "double_slash_redirecting"    "Prefix_Suffix"
##  [7] "having_Sub_Domain"           "SSLfinal_State"
```

```
##  [9] "Domain_registeration_length" "Favicon"
## [11] "port"                        "HTTPS_token"
## [13] "Request_URL"                 "URL_of_Anchor"
## [15] "Links_in_tags"               "SFH"
## [17] "Submitting_to_email"         "Abnormal_URL"
## [19] "Redirect"                    "on_mouseover"
## [21] "RightClick"                  "popUpWidnow"
## [23] "Iframe"                      "age_of_domain"
## [25] "DNSRecord"                   "web_traffic"
## [27] "Page_Rank"                   "Google_Index"
## [29] "Links_pointing_to_page"      "Statistical_report"
## [31] "Result"
```

- The data set is filled with integers all of which are either 1,0 or -1. These integers represent whether the method that was used to predict which websites are malicious(phishing). Therefore, 1 stands for legitimate website, 0 stands for suspicious website and -1 stands for phishing website.

```
str(df) # Quick description of the dataset;

## 'data.frame': 999 obs. of  31 variables:
##  $ having_IP_Address          : int  -1 1 1 1 1 -1 1 1 1 1 ...
##  $ URL_Length                 : int  1 1 0 0 0 0 0 0 0 1 ...
##  $ Shortining_Service         : int  1 1 1 1 -1 -1 -1 1 -1 -1 ...
##  $ having_At_Symbol           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ double_slash_redirecting   : int  -1 1 1 1 1 -1 1 1 1 1 ...
##  $ Prefix_Suffix              : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ having_Sub_Domain          : int  -1 0 -1 -1 1 1 -1 -1 1 -1 ...
##  $ SSLfinal_State             : int  -1 1 -1 -1 1 1 -1 -1 1 1 ...
##  $ Domain_registeration_length: int  -1 -1 -1 1 -1 -1 1 1 -1 -1 ...
##  $ Favicon                    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ port                       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ HTTPS_token                : int  -1 -1 -1 -1 1 -1 1 -1 -1 1 ...
##  $ Request_URL                : int  1 1 1 -1 1 1 -1 -1 1 1 ...
##  $ URL_of_Anchor              : int  -1 0 0 0 0 0 -1 0 0 0 ...
##  $ Links_in_tags              : int  1 -1 -1 0 0 0 0 -1 1 1 ...
##  $ SFH                        : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ Submitting_to_email        : int  -1 1 -1 1 1 -1 -1 1 1 1 ...
##  $ Abnormal_URL               : int  -1 1 -1 1 1 -1 -1 1 1 1 ...
##  $ Redirect                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ on_mouseover               : int  1 1 1 1 -1 1 1 1 1 1 ...
##  $ RightClick                 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ popUpWidnow                : int  1 1 1 1 -1 1 1 1 1 1 ...
##  $ Iframe                     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ age_of_domain              : int  -1 -1 1 -1 -1 1 1 -1 1 1 ...
##  $ DNSRecord                  : int  -1 -1 -1 -1 -1 1 -1 -1 -1 -1 ...
##  $ web_traffic                : int  -1 0 1 1 0 1 -1 0 1 0 ...
##  $ Page_Rank                  : int  -1 -1 -1 -1 -1 -1 -1 -1 1 -1 ...
##  $ Google_Index               : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Links_pointing_to_page     : int  1 1 0 -1 1 -1 0 0 0 0 ...
##  $ Statistical_report         : int  -1 1 -1 1 1 -1 -1 1 1 1 ...
##  $ Result                     : int  -1 -1 -1 -1 1 1 -1 -1 1 -1 ...
```

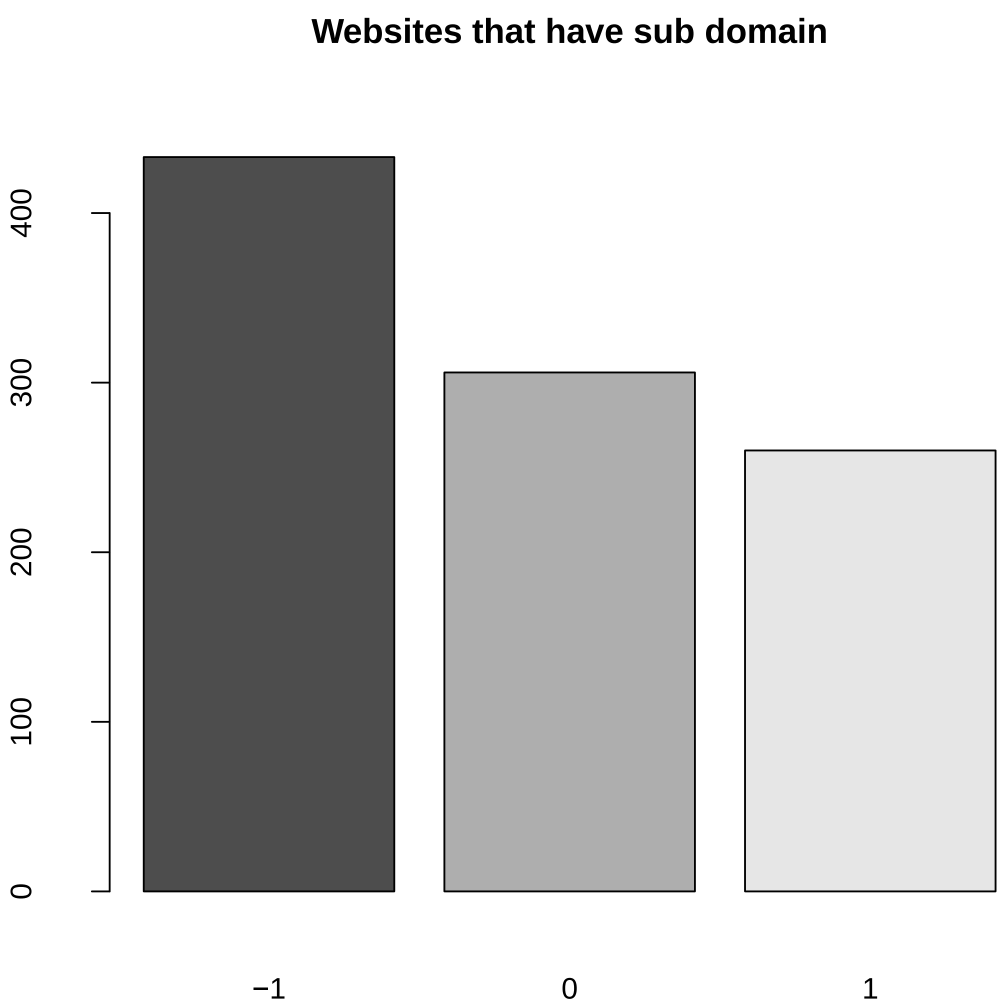
As we can see the distribution from the bar plot about 30 percent of the tested websites that have sub domains(see column -1) contain harmful malware, slighty more than 30 percent (see column 0) are suspicious and could harm the user and just over 40 percent are legitemate(see column 1). In addition, all numbers are integers, none are factors.

An example is given below of how the different techniques work to determine whether a website is phishing or legitemate. Since the data set consists of 31 attributes I thought it will be redundant if I explain how every single one works.

Attribute - having_IP_Address

If an IP address is used as an alternative of the domain name in the URL, such as url http://125.98.3.123/fake.html, users can be sure that someone is trying to steal their personal information. Sometimes, the IP address is even transformed into hexadecimal code as shown in the following link url http://0x58.0xCC.0xCA.0x62/2/paypal.ca/index.html. Rule: If The Domain Part has an IP Address - Phishing, Otherwise - Legitimate

```
par(mar=c(7.1, 7.1, 7.1, 2.1),mgp=c(3, 2, 0),las=0) #sets graphic grid to 2 by 2 with margin spacing
labelFreqs <- table(df$having_Sub_Domain) # frequency of Result
barplot(labelFreqs,col = grey.colors(3), # Shows the frequency of having sub domain;
        main="Websites that have sub domain")
```




```
#An example of results using a technique that checks whether the websites have sub domain
```

1.3. Pre-processing

```
colSums(is.na(df)) # check for missing values
```

```
##            having_IP_Address                   URL_Length
##                            0                            0
##           Shortining_Service              having_At_Symbol
##                            0                            0
##     double_slash_redirecting                Prefix_Suffix
##                            0                            0
##            having_Sub_Domain               SSLfinal_State
##                            0                            0
```

```
## Domain_registeration_length                        Favicon
##                            0                             0
##                         port                    HTTPS_token
##                            0                             0
##                  Request_URL                  URL_of_Anchor
##                            0                             0
##                Links_in_tags                            SFH
##                            0                             0
##          Submitting_to_email                   Abnormal_URL
##                            0                             0
##                     Redirect                   on_mouseover
##                            0                             0
##                   RightClick                    popUpWidnow
##                            0                             0
##                       Iframe                  age_of_domain
##                            0                             0
##                    DNSRecord                    web_traffic
##                            0                             0
##                    Page_Rank                   Google_Index
##                            0                             0
##      Links_pointing_to_page             Statistical_report
##                            0                             0
##                       Result
##                            0
```

- From the output above we can conclude that there are no missing values in the dataset. Therefore, techniques to predict the missing values are not required
- Standarise or normalise

```r
#standardize and normalise
dat <- data.frame(x = rnorm(10, 30, .2), y = runif(10, 3, 5))
scaled.dat <- scale(dat)

# check that we get mean of 0 and sd of 1
colMeans(scaled.dat)  # a version of apply(scaled.dat, 2, mean)

##                          x                          y
##  0.0000000000000002498002 -0.0000000000000025374668

apply(scaled.dat, 2, sd)

## x y
## 1 1
```

From the results above we can conclude that the mean is 1 and so is the standard deviation, thus the data values are equal or really close to the mean which is easily observable since the data only consists of -1,0, and 1.

Generate or drop features

Since the data is really well structured with no redundancy, there is no need to generate new features or to drop any of the current ones.

2 Modelling/Classification

- Vital part of modelling is to separate the given dataset into for example, one training and one testing subset. In this section, I divide the data into a training and a testing subsets, build a model with the training set and finally test,evaluate and discuss the results.

2.1. Divide data

```
library(dplyr)
library(caret)
library(ISLR)

idTrain <- createDataPartition(df$Result, p=0.7,list=FALSE, times=1)
# dividing the data 70% training set and 30% test set
train <- df[idTrain,] # training set with p = 0.7
sid <- as.numeric(rownames(train))
test <- df[-sid,] # test set with p = 0.3
```

2.2. Test and Evaluate

```
set.seed(99) #set seed for reproducibility
library(randomForest) # using randomforest libarary
prop.table(table(train$Result))

##
##        -1         1
## 0.4414286 0.5585714

ctrl = trainControl(method = "cv", number = 5) #using cross validation to go through the dataset which is separated
train$Result = as.factor(train$Result)# convert the numbers to factors
df$Result = ifelse(df$Result == -1,0,1) # make the numbers be only 0 and 1(instead of -1 and 1)
df$Result = as.factor(df$Result) # convert the numbers to factors in the main dataframe
fitModel <- train(Result~., #create a model
                  data=train, # data is set to be equal to train
                  method="rf", #using randomforest algorithm
                  trControl=ctrl,
                  ntree = 5) # number of trees

pred <- predict(fitModel, test[,-ncol(test)]) #predict that dataset
test <- cbind(test,pred) #bind predictions
results <- confusionMatrix(table(test$pred, test$Result))
accuracy <- sum(diag(results$table))/nrow(test)
cat('Accuracy is ', accuracy) # show results

## Accuracy is  0.9063545
```

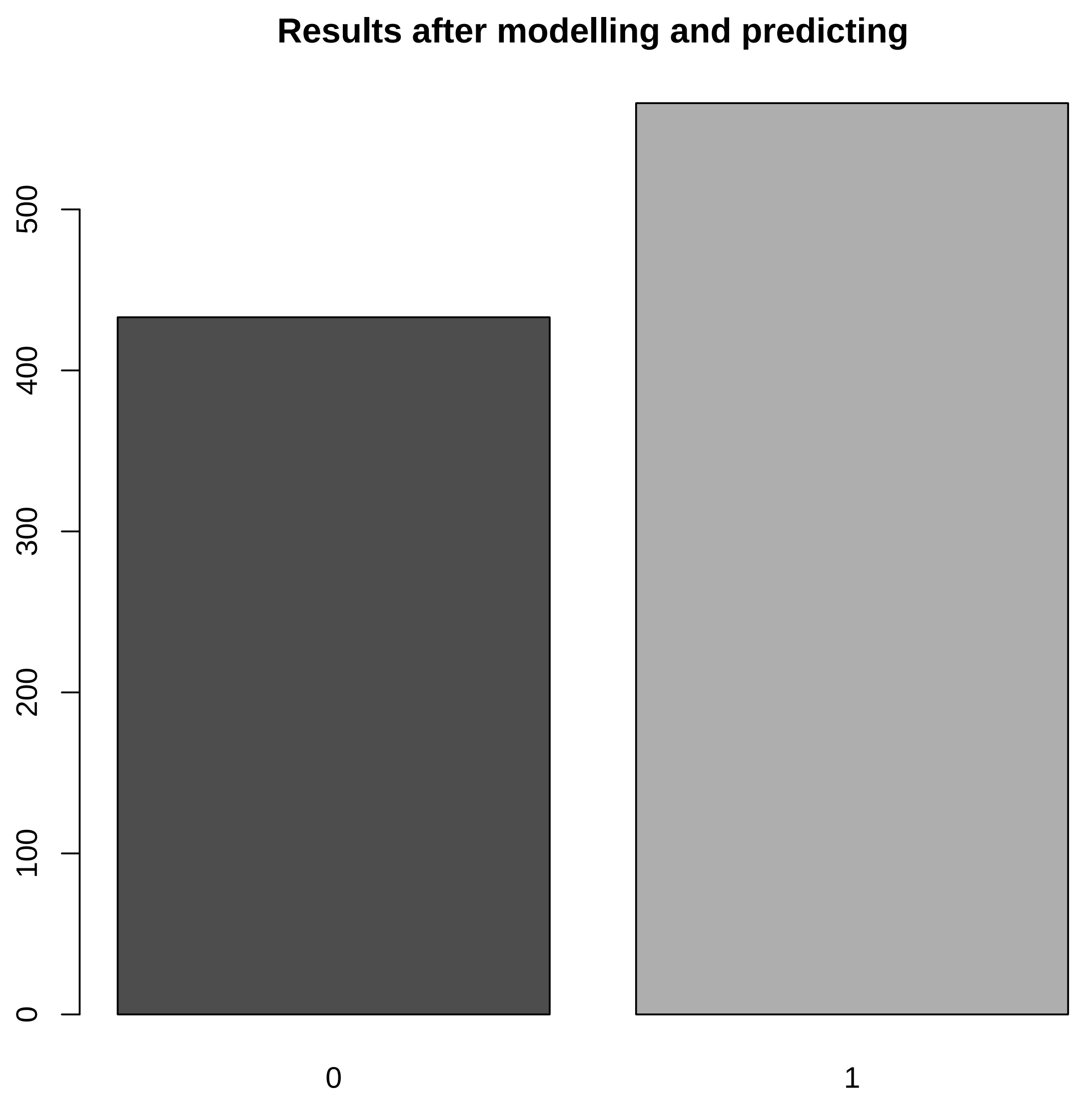
As a class variable we use 'Result' which contains only 1 and -1, where 1 stands for legitimate website and -1 for phishing website.

2.3. Report and discuss results

```
#Illustration and explaination .......
print(fitModel)
```

```
## Random Forest
##
## 700 samples
##  30 predictor
##   2 classes: '-1', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 560, 560, 560, 560, 560
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8671429  0.7271435
##   16    0.9085714  0.8149341
##   30    0.9185714  0.8351717
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 30.
```

```
aLabel <- table(df$Result) # frequency of Result
barplot(aLabel,col = grey.colors(3),
        main="Results after modelling and predicting")
```



From the barplot above we can observe that the dataset is balanced and we are able to predict whether a website contains malware with approximately ninety percent accuracy.

3 Improving performance

3.1. Tune parameters change partitioning of the dataset

```
idTrainTwo <- createDataPartition(df$Result, p=0.8,list=FALSE, times=1)
# this time we are dividing the data into a 80% training set and a 20% test set

trainTwo <- df[idTrainTwo,] # training set with p = 0.8
sid <- as.numeric(rownames(trainTwo))
testTwo <- df[-sid,] # test set with p = 0.2
set.seed(201)
```

```r
prop.table(table(trainTwo$Result))

##
##       0       1
## 0.43375 0.56625
```

3.2. Different models

```r
ctrlTwo = trainControl(method = "cv", number = 10)
# increasing the number variable by 5 (the previous model in section two is just 5)
trainTwo$Result = as.factor(trainTwo$Result)
fitModelTwo <- train(Result~.,
                  data=trainTwo,
                  method="rf", # again with random forest
                  trControl=ctrlTwo,
                  ntree=201)
# increasing the number of trees to be with 196 more than the one in the previous model

predTwo <- predict(fitModelTwo, testTwo[,-ncol(testTwo)]) #predict using the new model
testTwo <- cbind(testTwo,predTwo)
resultsTwo <- confusionMatrix(table(testTwo$predTwo, testTwo$Result))
accuracyTwo <- sum(diag(resultsTwo$table))/nrow(testTwo)
cat('Accuracy is ', accuracyTwo)

## Accuracy is  0.9497487

print(fitModelTwo)

## Random Forest
##
## 800 samples
##  30 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 720, 720, 720, 721, 720, 720, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9275733  0.8517811
##   16    0.9274953  0.8523839
##   30    0.9237920  0.8447274
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

We can observe that the accuracy has increased slightly by making the changes we made. Increasing the number of decision trees(from 5 to 201) helps to increase the precision of the algorithm.

Conclusion

The above illustrates that the random forest manages to predict extremely well whether a website contains malware with approximately 90 percent accuracy. Increasing the number of decision trees and cross validation split number increases the accuracy slightly.