LinkPrediction

Zahra Khoshmanesh

5/15/2020

```
library(linkprediction)
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 3.6.3
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4
```

```
## Warning: package 'tidyr' was built under R version 3.6.3
```

```
## Warning: package 'purrr' was built under R version 3.6.3
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
## Warning: package 'forcats' was built under R version 3.6.3
```

```
## -- Conflicts -------------------------------------------------------------------------------------
## x dplyr::as_data_frame() masks tibble::as_data_frame(), igraph::as_data_frame()
## x purrr::compose()       masks igraph::compose()
## x tidyr::crossing()      masks igraph::crossing()
## x dplyr::filter()        masks stats::filter()
## x dplyr::groups()        masks igraph::groups()
## x dplyr::lag()           masks stats::lag()
## x purrr::simplify()      masks igraph::simplify()
```

```
library(dplyr)
library(lattice)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(C50)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(mlbench)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(caretEnsemble)
```

```
##
## Attaching package: 'caretEnsemble'

## The following object is masked from 'package:ggplot2':
##
##     autoplot
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(klaR)
```

```
## Warning: package 'klaR' was built under R version 3.6.3
```

```
library(nnet)
```

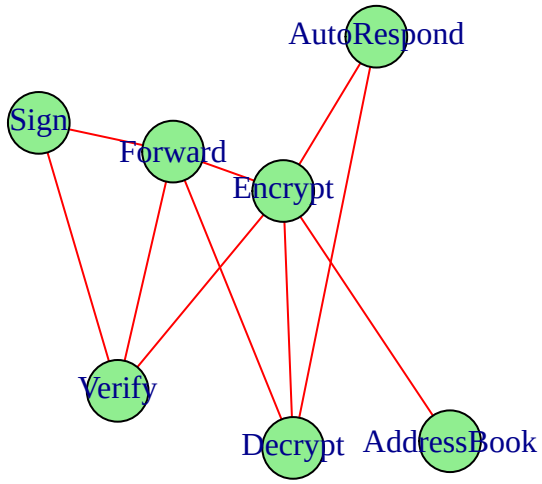
Rq3: What similarity metrics do perform better in the context of feature interaction detection?

creating Email graph

```
#creating graph of interactions
el_name <-  matrix(c("Decrypt","Forward",
                     "AddressBook","Encrypt",
                     "Sign","Verify",
                     "Sign","Forward",
                     "Encrypt","Decrypt",
                     "Encrypt","Verify",
                     "Encrypt","AutoRespond",
                     "Encrypt","Forward",
                     "Decrypt","AutoRespond",
                     "Verify","Forward"), nrow = 10, ncol = 2, byrow = TRUE)

interaction_graph <- graph_from_edgelist(el_name,directed = FALSE)
interaction_graph
```

```
## IGRAPH 8a217e6 UN-- 7 10 --
## + attr: name (v/c)
## + edges from 8a217e6 (vertex names):
##  [1] Decrypt    --Forward     AddressBook--Encrypt     Sign       --Verify
##  [4] Forward    --Sign        Decrypt    --Encrypt     Encrypt    --Verify
##  [7] Encrypt    --AutoRespond Forward    --Encrypt     Decrypt    --AutoRespond
## [10] Forward    --Verify
```

```
plot(interaction_graph, layout=layout_with_graphopt, vertex.color="lightgreen",edge.color="red",vertex.s
```



create unwanted feature interaction matrix of the Email Software Product Line

Decrypt: 1 Forward: 2 AddressBook: 3 Encrypt: 4 Sign: 5 Verify: 6 AutoRespond: 7

```
#create unwanted feature interaction matrix of email system

       fi <- matrix(c(1,2,
                      3,4,
                      5,6,
                      2,5,
                      1,4,
                      4,6,
                      4,7,
                      2,4,
```

```
                      1,7,
                      2,6,
                      1,3,#non fi nodes starts here
                      1,5,
                      1,6,
                      2,3,
                      2,7,
                      3,5,
                      3,6,
                      3,7,
                      4,5,
                      5,7,
                      6,7
                      ), nrow = 21, ncol = 2, byrow = TRUE)

#print and check the graph

fi
```

```
##       [,1] [,2]
##  [1,]    1    2
##  [2,]    3    4
##  [3,]    5    6
##  [4,]    2    5
##  [5,]    1    4
##  [6,]    4    6
##  [7,]    4    7
##  [8,]    2    4
##  [9,]    1    7
## [10,]    2    6
## [11,]    1    3
## [12,]    1    5
## [13,]    1    6
## [14,]    2    3
## [15,]    2    7
## [16,]    3    5
## [17,]    3    6
## [18,]    3    7
## [19,]    4    5
## [20,]    5    7
## [21,]    6    7
```

```
#create graph out of unwanted feature interaction matrix for email, the matrix is undirected
g <- graph_from_edgelist(fi,directed = FALSE)
#check the output of graph
g
```

```
## IGRAPH 8ae24d8 U--- 7 21 --
## + edges from 8ae24d8:
##  [1] 1--2 3--4 5--6 2--5 1--4 4--6 4--7 2--4 1--7 2--6 1--3 1--5 1--6 2--3 2--7
## [16] 3--5 3--6 3--7 4--5 5--7 6--7
```

```
#fi attributes shows which eges contributes to feature interactions

g <-set.edge.attribute(g, "fi", value=c("TRUE", "TRUE","TRUE","TRUE","TRUE","TRUE", "TRUE","TRUE","TRUE
                                        "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"

#set p1 as unwanted feature interactions exists in version 1 of the email product line,
g <-set.edge.attribute(g, "p1",  value=c("TRUE", "TRUE","TRUE","TRUE","TRUE","TRUE", "TRUE","TRUE","TRU
                                        "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"

#set p2 as unwanted feature interactions exists in version 2 of the email product line
g <-set.edge.attribute(g, "p2", value=c("TRUE", "TRUE","TRUE","TRUE","TRUE","TRUE", "TRUE","TRUE","TRUE
                                        "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"

get.edge.attribute(g, "fi")
```

```
##  [1] "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"
## [10] "TRUE"  "FALSE" "FALSE" "FALSE" "FALSE" "FALSE" "FALSE" "FALSE" "FALSE"
## [19] "FALSE" "FALSE" "FALSE"
```

```
get.edge.attribute(g, "p1")
```

```
##  [1] "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"
## [10] "TRUE"  "FALSE" "FALSE" "FALSE" "FALSE" "FALSE" "FALSE" "FALSE" "FALSE"
## [19] "FALSE" "FALSE" "FALSE"
```

```
get.edge.attribute(g, "p2")
```

```
##  [1] "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"
## [10] "TRUE"  "FALSE" "FALSE" "FALSE" "FALSE" "FALSE" "FALSE" "FALSE" "FALSE"
## [19] "FALSE" "FALSE" "FALSE"
```

Detection of each unwanted feature interaction based on other unwanted feature interactions

```
FI_detection <- function(g){

  ####################################################
  train <- delete.edges(g, which(E(g)$p1==FALSE))
  ####################################################
  aa <- proxfun(train, method="aa", value="edgelist") %>% dplyr::filter(from < to) %>% dplyr::rename(aa=

  #pa <- proxfun(train, method="pa", value="edgelist") %>%  filter(from < to) %>%  rename(pa=value)

  cosi <- proxfun(train, method="cos", value="edgelist") %>% dplyr:: filter(from < to) %>% dplyr::renam

  cn <- proxfun(train, method="cn", value="edgelist") %>%  dplyr::filter(from < to) %>%  dplyr::rename(

  jaccard <- proxfun(train, method="jaccard", value="edgelist") %>%  dplyr::filter(from < to) %>% dplyr
```

```
  ra <- proxfun(train, method="ra", value="edgelist") %>%  dplyr::filter(from < to) %>%  dplyr::rename(

  #global similarity metrics

  katz <- proxfun(train, method="katz", value="edgelist") %>%  dplyr::filter(from < to) %>%  dplyr::rena

  #act <- proxfun(train, method="act", value="edgelist") %>%  filter(from < to) %>%  rename(act=value)

  lp <- proxfun(train, method="lp", value="edgelist") %>%  dplyr::filter(from < to) %>% dplyr:: rename(

  rwr <- proxfun(train, method="rwr", value="edgelist") %>%  dplyr::filter(from < to) %>%  dplyr::renam

  ##################################################

  #"true" edges from period 2
p2g<- igraph::as_data_frame(g, what="edges") %>% dplyr::as_tibble() %>% dplyr::filter(p2==TRUE)
##################################################

testdf <- tidyr::crossing(
  # All dyads -- all pairs of vertex ids
  from = seq(1, vcount(train)),
  to = seq(1, vcount(train))
  ) %>%
  # The network is undirected thus we keep
  # only unique unordered pairs of vertex ids
    dplyr::filter(from < to) %>%
  # Join "true" edges from period 2
  dplyr::left_join(p2g,  by = c("from", "to")  ) %>%
  # Dyads without a match (have NAs) are disconnected
  # so we convert NAs to FALSE
  mutate_at(
  c("fi","p1", "p2"),
  function(x) ifelse(is.na(x), FALSE, x)
  ) %>%
  # Create logical variable `test` to flag new co-authorships.
  # These are present in `p2` but absent in `p1`.
  mutate(
  test = (p2==TRUE & p1==FALSE)# new co-authorships
  )
  ##########################################################
  testdf <- testdf %>% filter(p1==FALSE)
  ##########################################################
 preds <- testdf %>%
   left_join(aa, by=c("from", "to")) %>%
   #left_join(pa, by=c("from", "to")) %>%
   left_join(cosi, by=c("from", "to")) %>%
   left_join(cn, by=c("from", "to")) %>%
   left_join(jaccard, by=c("from", "to")) %>%
   left_join(ra, by=c("from", "to")) %>%
   left_join(katz, by=c("from", "to")) %>%
   #left_join(act, by=c("from", "to")) %>%
   left_join(lp, by=c("from", "to")) %>%
   left_join(rwr, by=c("from", "to")) %>%
```

```
  mutate_at(
  c("aa", "cosi","cn","jaccard","ra","katz","lp","rwr"), funs(ifelse(is.na(.), 0, .))
  )

  ##############################################################
  library(ROCR, warn.conflicts = FALSE)
  predlist <- lapply(
  c("aa","cosi","cn","jaccard","ra","katz","lp","rwr"),
  function(n) prediction(preds[[n]], preds$test)
  )
  names(predlist) <- c("aa", "cosi","cn","jaccard","ra","katz","lp","rwr")
  perflist <- lapply(predlist, performance, "tpr", "fpr")


  pal <- RColorBrewer::brewer.pal(8, "Set1")
  for(i in seq(along=perflist)) {
  plot(
  perflist[[i]],
  col = pal[i],
  add = i != 1
  ) }
  abline(a=0, b=1, lty="dashed")
  legend(
  "bottomright",
  title = "Methods",
  legend = c("Adamic-Adar", "Cosine based L+","Common Neighbour","Jaccard","RA","katz","lp","rwr"),
  lty = 1,
  col = pal,
  bty = "n"
  )

  ##########################################
  vapply( predlist,function(p) performance(p, "auc")@y.values[[1]],  numeric(1) )

}
```

1–2 3–4 5–6 2–5 1–4 4–6 4–7 2–4 1–7 2–6

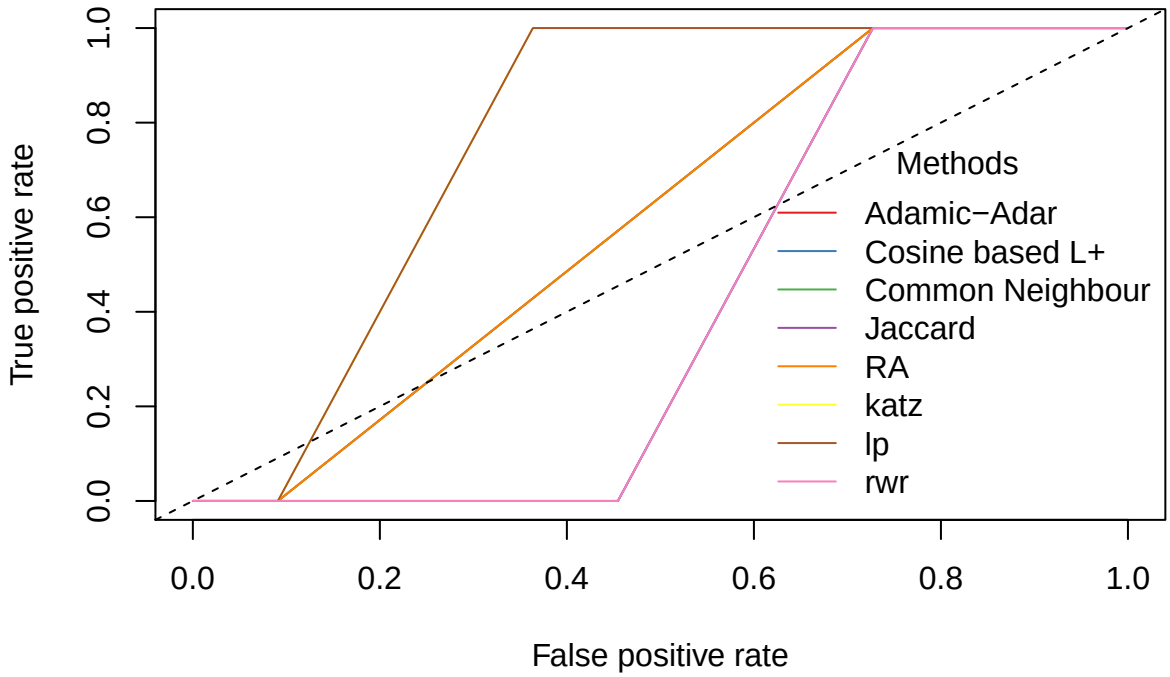
Decrypt: 1 Forward: 2 AddressBook: 3 Encrypt: 4 Sign: 5 Verify: 6 AutoRespond: 7

1- Detection of Uuwanted Feature Interaction (1,2): Decrypt-Forward

```
g <-set.edge.attribute(g, "p1",  value=c("FALSE", "TRUE","TRUE","TRUE","TRUE","TRUE", "TRUE","TRUE","TRU
                                         "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"
FI_detection(g)
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
```

```
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.

## Warning: package 'ROCR' was built under R version 3.6.3

## Loading required package: gplots

## Warning: package 'gplots' was built under R version 3.6.3

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```




```
##        aa      cosi        cn   jaccard        ra      katz        lp       rwr
## 0.5909091 0.4090909 0.5909091 0.4090909 0.5909091 0.7727273 0.7727273 0.4090909
```

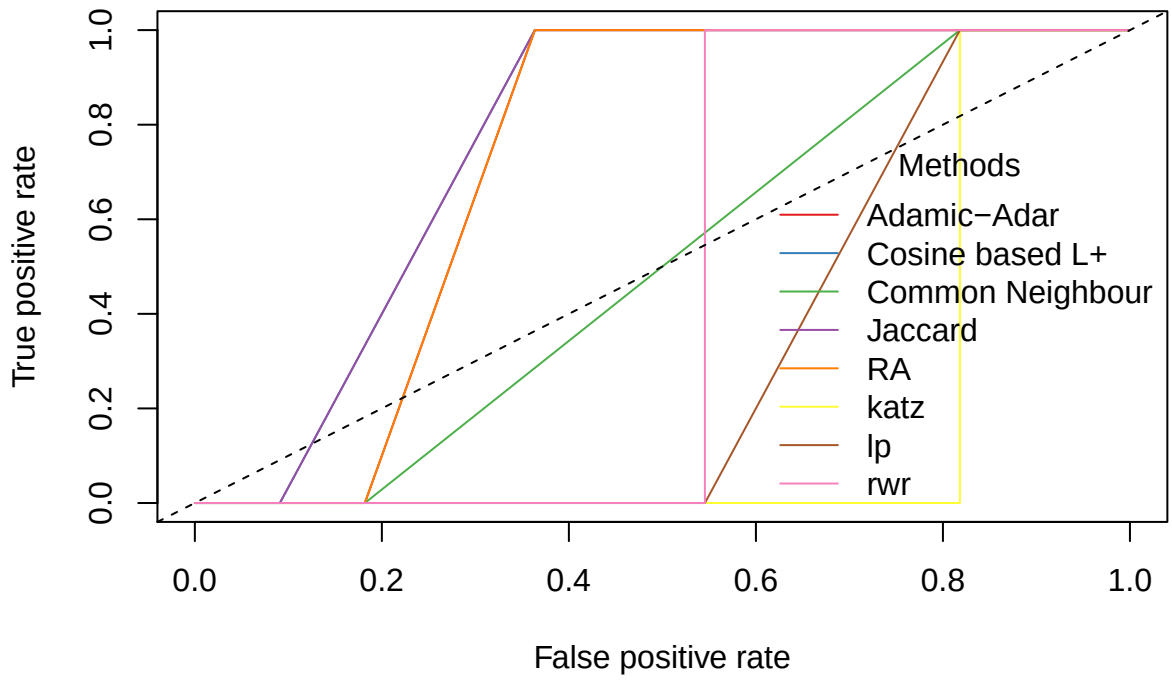
2- Detection of Uuwanted Feature Interaction (3,4): AddressBook-Encrypt

give error since the graph will must be connected

```
g <-set.edge.attribute(g, "p1",  value=c("TRUE", "FALSE","TRUE","TRUE","TRUE","TRUE", "TRUE","TRUE","TRU
                                         "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"
#FI_detection(g)
```

3- Detection of Uuwanted Feature Interaction (5,6): Sign-Verify

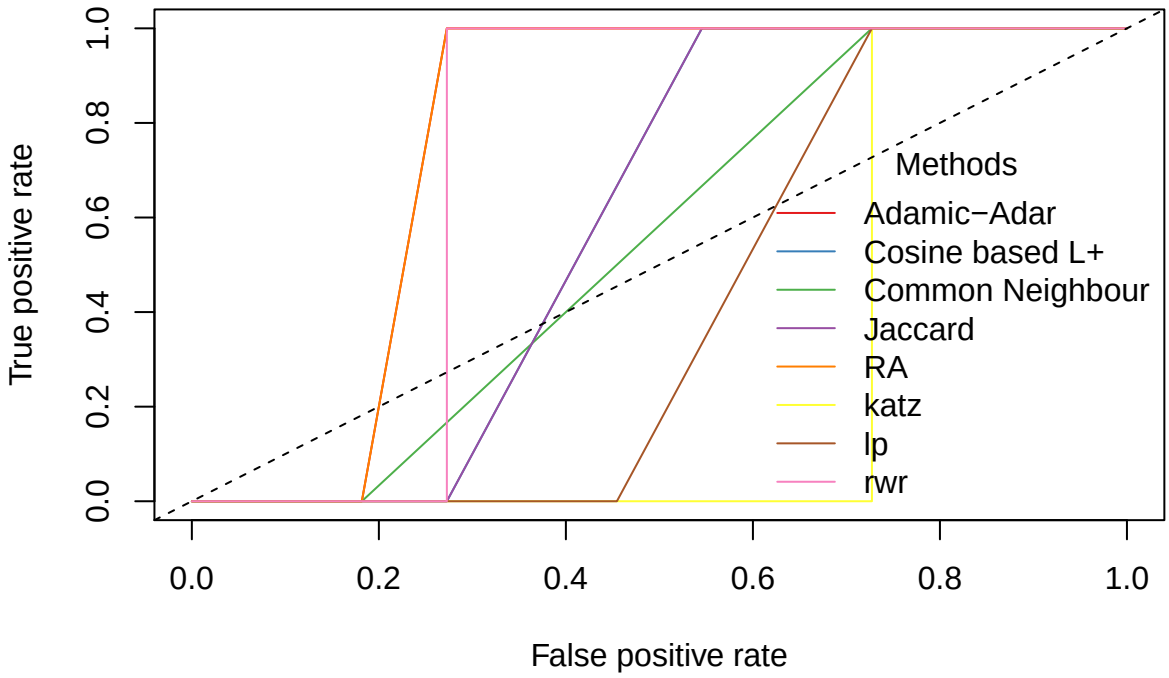
```
g <-set.edge.attribute(g, "p1",  value=c("TRUE", "TRUE","FALSE","TRUE","TRUE","TRUE", "TRUE","TRUE","TRU
                                         "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"
FI_detection(g)
```




```
##        aa      cosi        cn   jaccard        ra      katz        lp       rwr
## 0.7272727 0.7727273 0.5000000 0.7727273 0.7272727 0.1818182 0.3181818 0.4545455
```

4- Detection of Uuwanted Feature Interaction (2,5): Forward-Sign

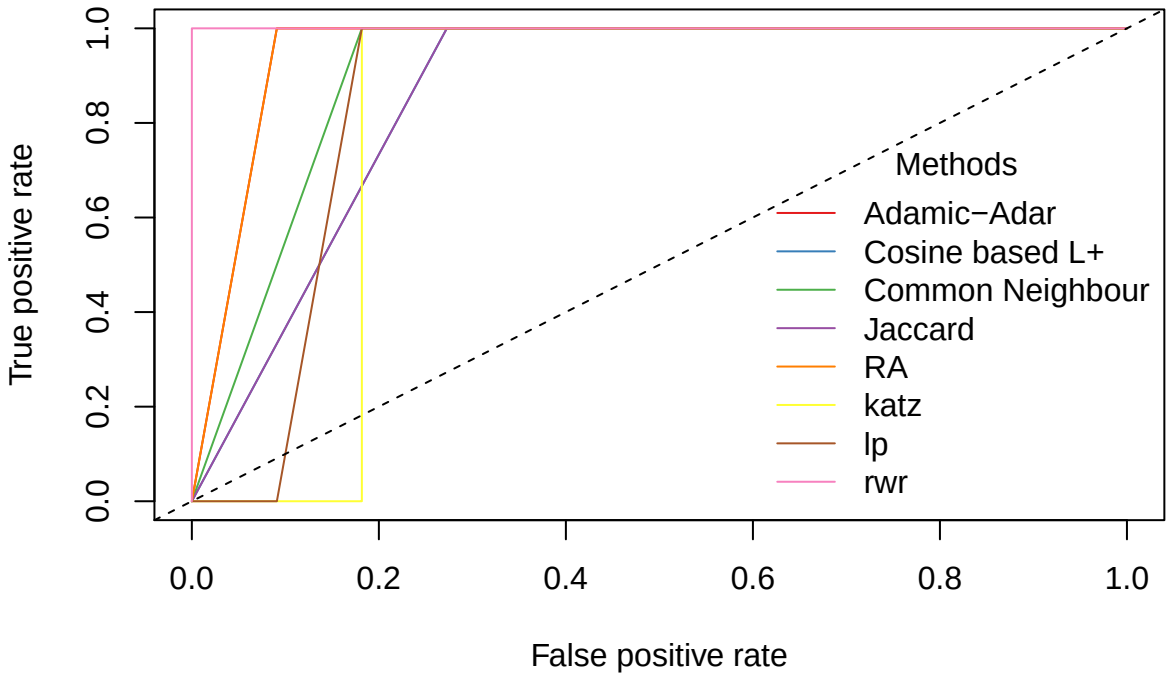
```
g <-set.edge.attribute(g, "p1",  value=c("TRUE", "TRUE","TRUE","FALSE","TRUE","TRUE", "TRUE","TRUE","TRU
                                         "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"
FI_detection(g)
```




```
##        aa      cosi        cn   jaccard        ra      katz        lp       rwr
## 0.7727273 0.5909091 0.5454545 0.5909091 0.7727273 0.2727273 0.4090909 0.7272727
```

5- Detection of Uuwanted Feature Interaction (1,4): Decrypt-Encrypt

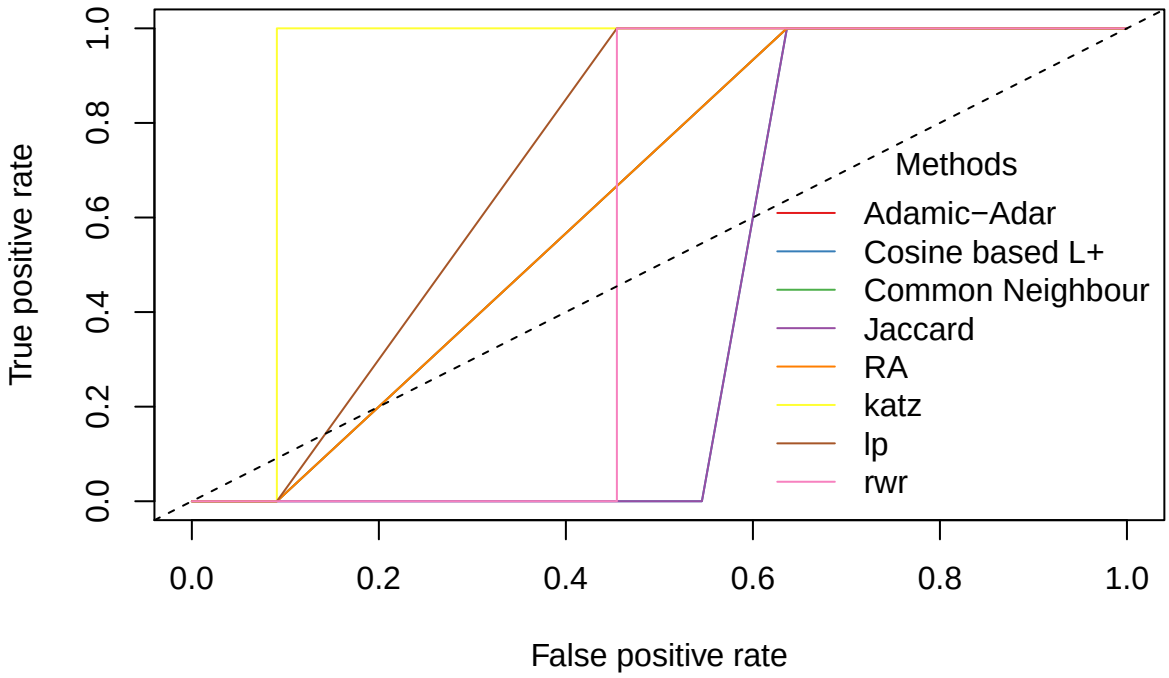
```
g <-set.edge.attribute(g, "p1",  value=c("TRUE", "TRUE","TRUE","TRUE","FALSE","TRUE", "TRUE","TRUE","TRU
                                         "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"
FI_detection(g)
```




```
##        aa      cosi        cn   jaccard        ra      katz        lp       rwr
## 0.9545455 0.8636364 0.9090909 0.8636364 0.9545455 0.8181818 0.8636364 1.0000000
```

6- Detection of Uuwanted Feature Interaction (4,6): Encrypt-Verify

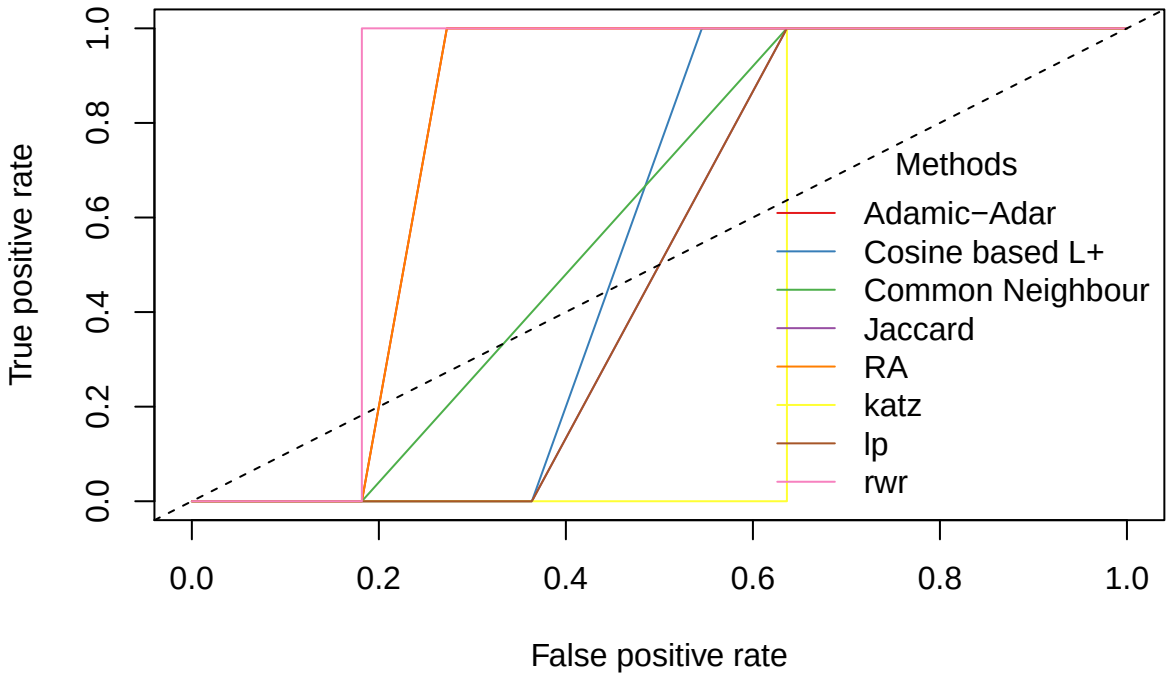
```
g <-set.edge.attribute(g, "p1",  value=c("TRUE", "TRUE","TRUE","TRUE","TRUE","FALSE", "TRUE","TRUE","TRU
                                         "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"
FI_detection(g)
```




```
##        aa      cosi        cn   jaccard        ra      katz        lp       rwr
## 0.6363636 0.4090909 0.6363636 0.4090909 0.6363636 0.9090909 0.7272727 0.5454545
```

7- Detection of Uuwanted Feature Interaction (4,7): Encrypt-AutoRespond

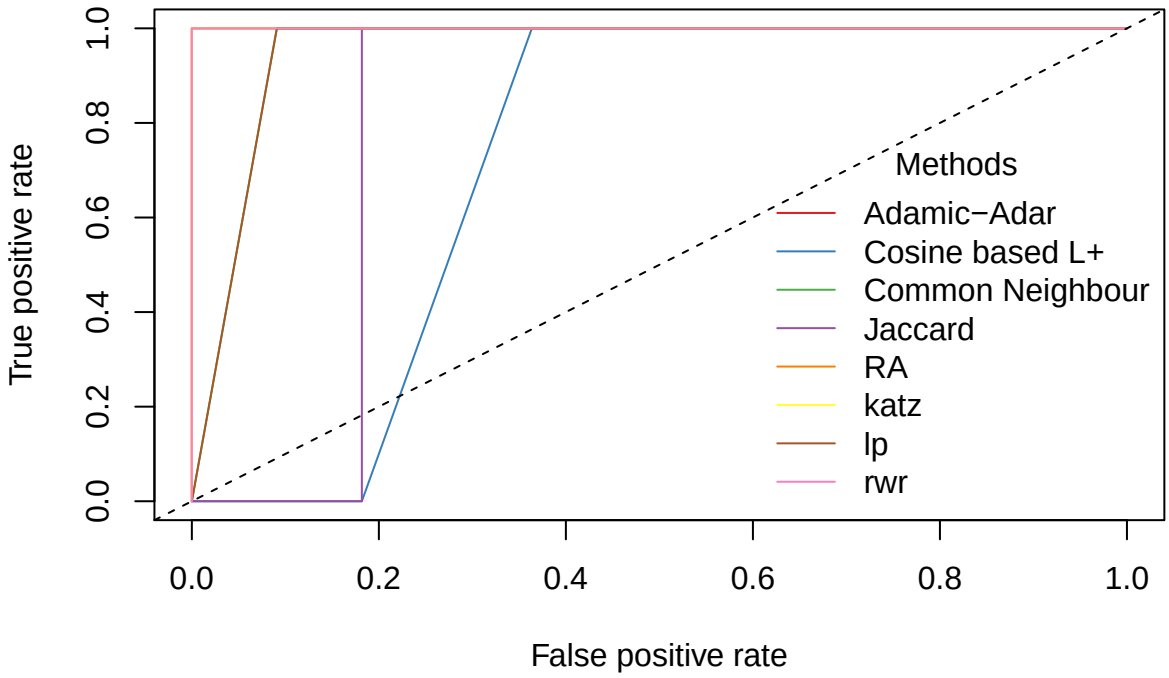
```
g <-set.edge.attribute(g, "p1",  value=c("TRUE", "TRUE","TRUE","TRUE","TRUE","TRUE", "FALSE","TRUE","TRU
                                         "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"
FI_detection(g)
```




```
##        aa      cosi        cn   jaccard        ra      katz        lp       rwr
## 0.7727273 0.5454545 0.5909091 0.5000000 0.7727273 0.3636364 0.5000000 0.8181818
```

8- Detection of Uuwanted Feature Interaction (2,4): Forward-Encrypt

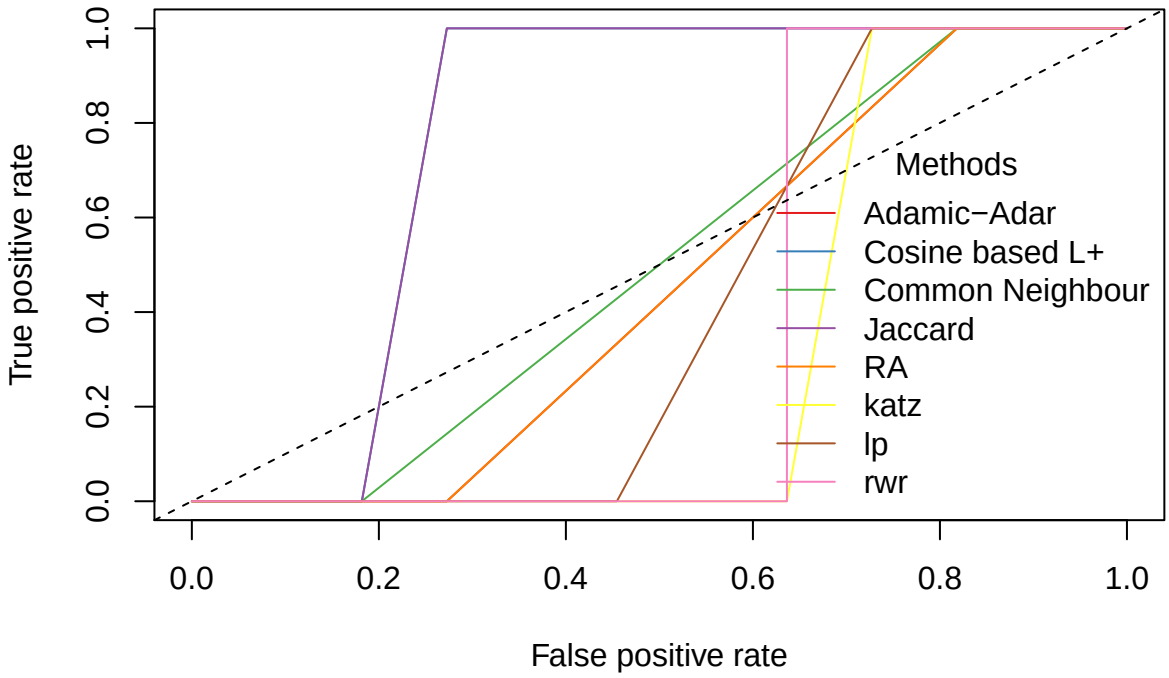
```
g <-set.edge.attribute(g, "p1",  value=c("TRUE", "TRUE","TRUE","TRUE","TRUE","TRUE", "TRUE","FALSE","TRU
                                         "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"
FI_detection(g)
```




```
##        aa      cosi        cn   jaccard        ra      katz        lp       rwr
## 1.0000000 0.7272727 0.9545455 0.8181818 1.0000000 1.0000000 0.9545455 1.0000000
```

9- Detection of Uuwanted Feature Interaction (1,7): Decrypt-AutoRespond

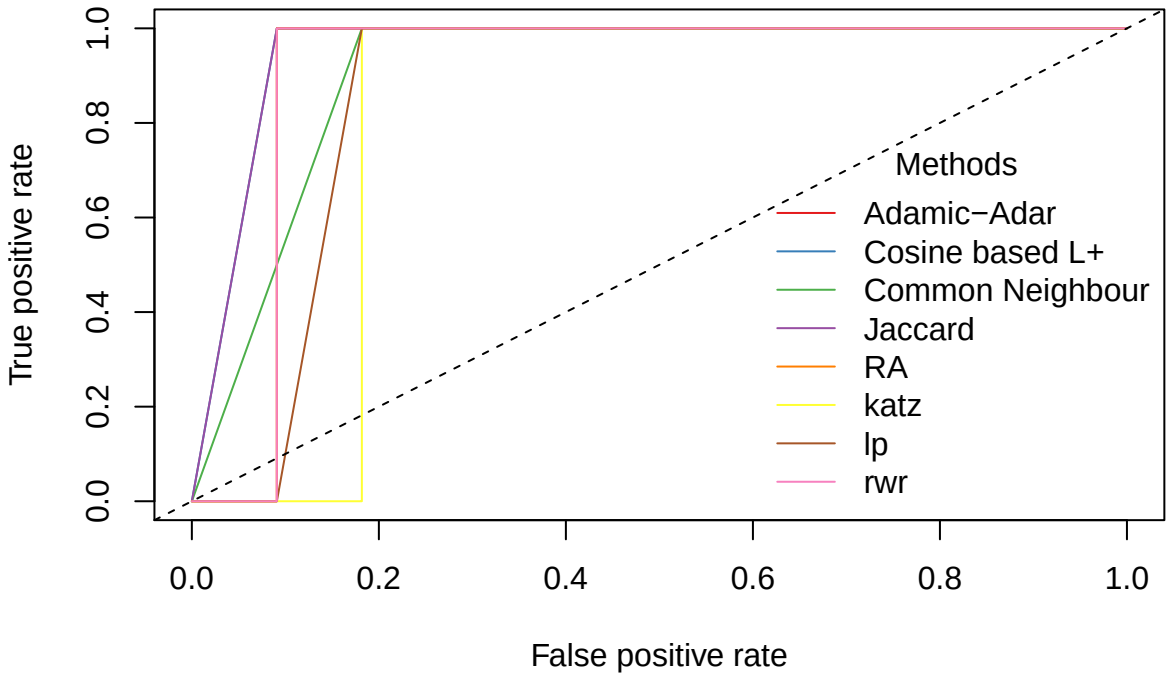
```
g <-set.edge.attribute(g, "p1",  value=c("TRUE", "TRUE","TRUE","TRUE","TRUE","TRUE", "TRUE","TRUE","FALS
                                         "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"
FI_detection(g)
```




```
##        aa      cosi        cn   jaccard        ra      katz        lp       rwr
## 0.4545455 0.7727273 0.5000000 0.7727273 0.4545455 0.3181818 0.4090909 0.3636364
```

10- Detection of Uuwanted Feature Interaction (2,6): Forward-Verify

```
g <-set.edge.attribute(g, "p1",  value=c("TRUE", "TRUE","TRUE","TRUE","TRUE","TRUE", "TRUE","TRUE","TRU
                                         "FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE","FALSE"
FI_detection(g)
```




```
##        aa      cosi        cn   jaccard        ra      katz        lp       rwr
## 0.9090909 0.9545455 0.9090909 0.9545455 0.9090909 0.8181818 0.8636364 0.9090909
```