Name: Chan Ka Tsun
SID: 53563363

Q1 (See Appendix III – Coding)

Data Summary: Hero

	day_diff_0	sum_price_rev1	day_diff_1	sum_price_rev2	day_diff_2	sum_price_rev3	purch_amount_total	purch_days_no_total
Whale (n = 171)	4.5	64,854 (n = 171)	4.2	79,053 (n = 171)	4.5	78,924 (n = 171)	Mean: 1,671,485 S.D.: Popn: 856,255 Sample: 858769	Mean: 22.7 S.D.: 11.3
Dolphin (n =1,933)	8.8	38,760 (n = 1,933)	7.1	49,235 (n = 1,827)	7.6	48,221 (n = 1671)	Mean: 294,428 S.D.: Popn: 205,751 Sample: 205804	Mean: 8.0 S.D.: 6.0
Minnow (n = 9,796)	13.1	6,344 (n = 9,796)	22.0	9,450 (n = 4,713)	21.1	8,550 (n = 2,623)	Mean: 15,811 S.D.: Popn: 20,825 Sample: 20826	Mean: 2.1 S.D.: 1.8
Total (n =11,900)	12.1	12,451 (n = 11,900)	17.5	22,055 (n = 6,711)	15.4	26,092 (n = 4,465)	Mean: 84,860 S.D.: 254,838 p1: 1,181,010 p10: 203,000 p50: 10,000	Mean: 3.4 S.D.: 4.5

Name: Chan Ka Tsun
SID: 53563363

Question 2: Variables generated/newly created (See Appendix III Coding Q2, Appendix II Var) Number of rows: 11900 (outer join) Number of columns: 30		
Category	**Var**	**Description**
	isandroid	Android user?
	dimension	Screen size (w*h)
	is_dolphin	
1st and 2nd purchasing record	count_report_value1	
	sum_report_value1	
	sum_price_rev1	
	daydiff1	
	count_report_value2	
	sum_report_value2	
	sum_price_rev2	
	daydiff2	
	sum_price_inc	Purchase increase from 1st to 2nd
	purch_date1	
Addictiveness	no_usage_before_purch	
	usage_total_before_purch	
	freq_usage_before_purch	
	session_mean_before_purch	
	session_var_before_purch	
	usage_increase_before_purch	
	session_max_before_purch	Longest session before purchase
	sess_date_diff_avg	Average no. of days between sessions
	no_connection_before_purch	
	connection_total_before_purch	
	freq_connection_before_purch	
	connection_mean_before_purch	
	connection_var_before_purch	
	connection_increase_before_purch	
	conn_date_diff_avg	Average no. of days between connections
	User_no	
	Install_date	

Name: Chan Ka Tsun
SID: 53563363

Q3.

Training dataset: 8303 rows (70%)

Testing dataset: 3597 rows (30%)

Seed: 5

Prediction result:

	Rptree	Citree	Logistic regression
Sensitivity	0.91	0.80	0.81
Specificity	0.78	0.75	0.89
AUC	0.89	0.87	0.92

Methodology

Steps taken:

Prepared the training and testing datasets

Built a classification model with recursive partitioning trees

Visualized a recursive partitioning tree

Measured the prediction performance of a recursive partitioning tree

Pruned a recursive partitioning tree

Built a classification model with a conditional inference tree

Visualized a conditional inference tree

Measured the prediction performance of a conditional inference tree

Classified data with logistic regression

Evaluate models from sensitivity, specificity, AUC, ROC etc.

Choose: Logistic Regression

We chose Logistic Regression model due to its various advantages and performance. Logistic regression is relatively easy to interpret, it also directs model logistic probability, and provides a confidence interval for the outcome. On the other hand, in decision tree it is hard to update the model, whereas, in logistic regression, the classification model can be updated to incorporate new data, i.e. keep updating in real time, which makes it faster to identify which one is dolphin user before they quit the game. The AUC is also the highest among all.

Recursive Partitioning tree (RP) chooses variables to maximize information gain, which is based on entropy level, while Conditional Inference tree (CI) adapts the significant test procedures to select variables. Although the sensitivity of RP is the highest among all, it suffers from overfitting problem and it is prone to bias. For the CI tree, it is prone to over fitting. More techniques such as a random forest method or tree pruning might be considered to solve the problem of overfitting.

Name: Chan Ka Tsun
SID: 53563363

Appendix I - Graphs

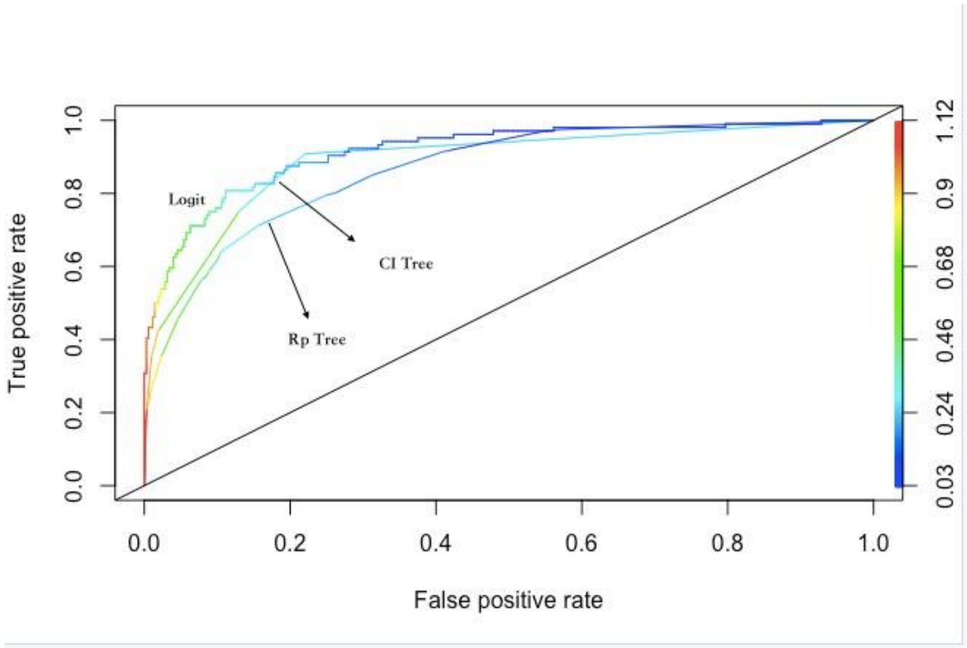

Figure 1 ROC

Figure 2 RP Tree

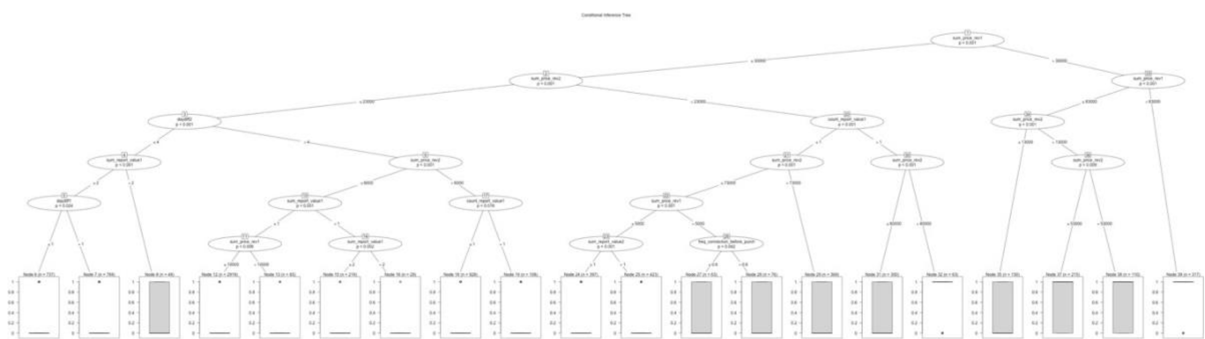

Figure 3 CI Tree

Name: Chan Ka Tsun
SID: 53563363

Appendix II - Variables

Q2

Number of rows: 11900 (outer join)

Number of columns: 30

User profile:

1. Screen size (width * height)
2. Is android
3. Is dolphin

Addictiveness:

1. Number of reported day usage before first purchase (Session)
2. Session total before first purchase
3. Frequency usage before first purchase
4. Session mean before first purchase
5. Session variance before first purchase
6. Usage increase before first purchase
7. Maximum session before first purchase
8. Session date difference
9. Number of day connection before first purchase
10. Total connection before first purchase
11. Connection frequency before first purchase
12. Connection mean before first purchase
13. Connection variance before first purchase
14. Connection increase before first purchase
15. Connection date difference

First and second purchasing record:

1. Count report value X
2. Sum report value X
3. Sum price rev X
4. Daydiff 1
5. Daydiff 2
6. Sum price increase (from purchase 1 to purchase 2)

Useless:

1. User no
2. Install date
3. First purchase date

Name: Chan Ka Tsun
SID: 53563363

Appendix III - Coding

Q1

Preprocessing the data in python

```
import pandas as pd

buysum = pd.read_csv("/Users/Lwmformula/Downloads/Asgn3/buy_activity_summary.csv",index_col=False)
buysum= buysum.drop(buysum.columns[0],axis=1,inplace=False)
buysum= buysum.drop(['no_purcha_days','no_purcha'],axis=1,inplace=False)

buyhist = pd.read_csv("/Users/Lwmformula/Downloads/Asgn3/buy_history.csv",index_col=False)
buyhist = buyhist.drop(buyhist.columns[0],axis=1,inplace=False)

ucd = pd.read_csv("/Users/Lwmformula/Downloads/Asgn3/usage_connection_daily.csv",index_col=False)
ucd = ucd.drop(ucd.columns[0],axis=1,inplace=False)

usd = pd.read_csv("/Users/Lwmformula/Downloads/Asgn3/usage_session_daily.csv",index_col=False)
usd = usd.drop(usd.columns[0],axis=1,inplace=False)

user = pd.read_csv("/Users/Lwmformula/Downloads/Asgn3/user.csv",index_col=False)
user = user.drop(user.columns[0],axis=1,inplace=False)
```

```
platform = user['platform_type'].tolist()
width = user['width'].tolist()
height = user['height'].tolist()
dimension = [width[i]*height[i] for i in range(len(width))]
isandroid = []
for i in platform:
    if i == 'android': isandroid.append(1)
    else: isandroid.append(0)
user['dimension'] = dimension
user['isandroid'] = isandroid
userinfo = user[['user_no','isandroid','dimension']]
```

Generating the summary table

```
import datetime
import numpy as np
import rpy2.robjects as ro
from rpy2.robjects import pandas2ri
import math

sumdf = buyhist[buyhist['app_key'] == 80862143]
sumdf = sumdf.drop(list(sumdf.columns[24:]),axis=1,inplace=False)
buysumt = buysum.drop(buysum.columns[2],axis=1,inplace=False)

insday = sumdf['install_date'].tolist()
purday1 = sumdf['purch_date1'].tolist()
purday2 = sumdf['purch_date2'].tolist()
purday3 = sumdf['purch_date3'].tolist()
daydiff1 = []
daydiff2 = []
daydiff3 = []

for i in range(len(insday)):
    d1 = datetime.datetime.strptime(purday1[i],"%Y-%m-%d")
    try:
        d0 = datetime.datetime.strptime(insday[i],"%Y-%m-%d")
        daydiff1.append((d1-d0).days)
    except:
        daydiff1.append(np.NaN)
    try:
        d2 = datetime.datetime.strptime(purday2[i],"%Y-%m-%d")
        daydiff2.append((d2-d1).days)
    except:
        daydiff2.append(np.NaN)
    try:
        d2 = datetime.datetime.strptime(purday2[i],"%Y-%m-%d")
        d3 = datetime.datetime.strptime(purday3[i],"%Y-%m-%d")
        daydiff3.append((d3-d2).days)
    except:
        daydiff3.append(np.NaN)
sumdf['daydiff1'] = daydiff1
```

Name: Chan Ka Tsun
SID: 53563363

```
sumdf['daydiff2'] = daydiff2
sumdf['daydiff3'] = daydiff3
sumdf = pd.merge(sumdf,buysumt,on=["app_key","user_no"])

classc = []
for i in sumdf['sum_expense'].tolist():
    if i >= 1000000: classc.append("whale")
    elif (i < 1000000) and (i >=100000): classc.append("dolphin")
    elif (i < 100000): classc.append('minnow')

sumdf['class'] = classc
'''
print sumdf.groupby(['daydiff1','daydiff2','daydiff3'])['daydiff1','sum_price_rev1',
                             'daydiff2','sum_price_rev2',
                             'daydiff3','sum_price_rev3'].agg(['mean','count'])


print np.nanmean(daydiff1)
print np.nanmean(sumdf['sum_price_rev1'])
print np.count_nonzero(~np.isnan(sumdf['sum_price_rev1']))

print np.nanmean(daydiff2)
print np.nanmean(sumdf['sum_price_rev2'])
print np.count_nonzero(~np.isnan(sumdf['sum_price_rev2']))

print np.nanmean(daydiff3)
print np.nanmean(sumdf['sum_price_rev3'])
print np.count_nonzero(~np.isnan(sumdf['sum_price_rev3']))
'''

#print sumdf.groupby('class')['purch_amount_total',
#                             'purch_days_no_total'].agg(['mean','std'])
'''
```

```
'''
print np.nanmean(sumdf['sum_expense'])
print np.nanstd(sumdf['sum_expense'])

print np.percentile(sumdf['sum_expense'],99)
print np.percentile(sumdf['sum_expense'],90)
print np.percentile(sumdf['sum_expense'],50)

print np.nanmean(sumdf['purch_days_no_total'])
print np.nanstd(sumdf['purch_days_no_total'])

#tmp = np.std(sumdf['sum_expense'].tolist())
#print math.sqrt((tmp**2*(11900))/(11900-1))

a = sumdf.groupby('class')['purch_amount_total'].std().tolist()[0]
b = sumdf.groupby('class')['purch_amount_total'].std().tolist()[1]
c = sumdf.groupby('class')['purch_amount_total'].std().tolist()[2]

d = sumdf.groupby('class')['purch_days_no_total'].std().tolist()[0]
e = sumdf.groupby('class')['purch_days_no_total'].std().tolist()[1]
f = sumdf.groupby('class')['purch_days_no_total'].std().tolist()[2]
print a,b,c
print d,e,f

print math.sqrt((a**2*(1933))/(1933-1))
print math.sqrt((b**2*(9796))/(9796-1))
print math.sqrt((c**2*(171))/(171-1))
print "========================================================================"
print math.sqrt((d**2*(1933))/(1933-1))
print math.sqrt((e**2*(9796))/(9796-1))
print math.sqrt((f**2*(171))/(171-1))

print math.sqrt((np.nanstd(sumdf['purch_days_no_total'])**2*(11900))/(11900-1))
'''
```

Name: Chan Ka Tsun
SID: 53563363

Q2

Preprocessing the data in python

```
import datetime
import math
def cal_inc(li):
    if len(li) == 1:
        return np.nan
    elif len(li)%2 != 0:
        fac = int(math.ceil(len(li)/2))
        a = sum(li[:fac+1])
        b = sum(li[fac:])
        return b/float(a)
    elif len(li)%2 == 0:
        fac = int(math.ceil(len(li)/2))
        a = sum(li[:fac])
        b = sum(li[fac-1:])
        return b/float(a)

is_dolphin = []
for i in sumdf['sum_expense'].tolist():
    if i >= 100000: is_dolphin.append(1)
    else: is_dolphin.append(0)
sumdf['is_dolphin'] = is_dolphin
b4df = sumdf[['user_no','install_date','purch_date1','is_dolphin','count_report_value1','sum_report_value1',
              'sum_price_rev1','daydiff1','count_report_value2','sum_report_value2','sum_price_rev2','daydiff2']]

price1 = b4df['sum_price_rev1'].tolist()
price2 = b4df['sum_price_rev2'].tolist()
sum_price_inc = [(price2[i]/float(price1[i])) for i in range(len(price1))]
b4df['sum_price_inc'] = sum_price_inc

tmpd = b4df['purch_date1'].tolist()
tmp_d = []
for i in tmpd:
    d = int(datetime.datetime.strptime(i,"%Y-%m-%d").strftime("%Y%m%d"))
    tmp_d.append(d)
b4df = b4df.drop("purch_date1",axis=1,inplace=False)
b4df['purch_date1'] = tmp_d
backup = b4df
```

Connection

```
b4df = pd.merge(b4df,ucd[ucd['app_key'] == 80862143],on="user_no")
#80862143
#10977901
b4fp = b4df[b4df['purch_date1'] > b4df['report_date']]
b4fp = b4fp.sort_values(['user_no','report_date'], ascending=[1, 1])

#index
cib4fp = b4fp.groupby('user_no').count()['report_value'].index.tolist()

#count
cub4fp = b4fp.groupby('user_no').count()['report_value'].tolist()

#frequency
daydiff0 = []
p = b4fp['purch_date1'].tolist()
ins = b4fp['install_date'].tolist()
for i in range(len(p)):
    try:
        pdd = datetime.datetime.strptime(str(p[i]),"%Y%m%d")
        idd = datetime.datetime.strptime(str(ins[i]),"%Y-%m-%d")
        daydiff0.append(((pdd-idd).days+1))
    except:
        daydiff0.append(np.nan)
b4fp['time_to_purch'] = daydiff0
tmp_ttp = b4fp.groupby('user_no')['time_to_purch'].apply(list).tolist()
tmp_rv = b4fp.groupby('user_no').count()['report_value'].tolist()
cfb4fp = []
for i in range(len(tmp_ttp)):
    try:
        cfb4fp.append(tmp_rv[i]/float(tmp_ttp[i][0]))
    except:
        cfb4fp.append(np.nan)
```

Name: Chan Ka Tsun
SID: 53563363

```python
#inc
tmpinlist = [i for i in b4fp.groupby('user_no')['report_value'].apply(list)]
cinc = []
for i in tmpinlist:
    cinc.append(cal_inc(i))

#connection_day_diff
conn_date = b4fp.groupby('user_no')['report_date'].apply(list).tolist()
conn_date_diff_avg = []
for i in conn_date:
    total = 0
    if len(i) == 1: conn_date_diff_avg.append(total)
    else:
        for j in range(len(i)):
            try:
                first = datetime.datetime.strptime(str(i[j]),"%Y%m%d")
                second = datetime.datetime.strptime(str(i[j+1]),"%Y%m%d")
                total += ((second - first).days)
            except:
                break
        conn_date_diff_avg.append((total/float(len(i))))

#sum,mean,var
csb4fp = b4fp.groupby('user_no').sum()['report_value'].tolist()
cmb4fp = b4fp.groupby('user_no').mean()['report_value'].tolist()
cvb4fp = b4fp.groupby('user_no').var()['report_value'].tolist()
```

Session

```python
b4df = backup
b4df = pd.merge(b4df,usd[usd['app_key'] == 80862143],on="user_no")
#80862143
#10977901

b4fp = b4df[b4df['purch_date1'] > b4df['report_date']]
b4fp = b4fp.sort_values(['user_no','report_date'], ascending=[1, 1])

#index
sib4fp = b4fp.groupby('user_no').count()['report_value'].index.tolist()

#count
sub4fp = b4fp.groupby('user_no').count()['report_value'].tolist()

#frequency
daydiff0 = []
p = b4fp['purch_date1'].tolist()
ins = b4fp['install_date'].tolist()
for i in range(len(p)):
    try:
        pdd = datetime.datetime.strptime(str(p[i]),"%Y%m%d")
        idd = datetime.datetime.strptime(str(ins[i]),"%Y-%m-%d")
        daydiff0.append(((pdd-idd).days+1))
    except:
        daydiff0.append(np.nan)
b4fp['time_to_purch'] = daydiff0
tmp_ttp = b4fp.groupby('user_no')['time_to_purch'].apply(list).tolist()
tmp_rv = b4fp.groupby('user_no').count()['report_value'].tolist()

sfb4fp = []
for i in range(len(tmp_ttp)):
    try:
        sfb4fp.append(tmp_rv[i]/float(tmp_ttp[i][0]))
    except:
        sfb4fp.append(np.nan)
```

Name: Chan Ka Tsun
SID: 53563363

```
#inc
tmpinlist = [i for i in b4fp.groupby('user_no')['report_value'].apply(list)]
sinc = []
for i in tmpinlist:
    sinc.append(cal_inc(i))

#session_day_diff
sess_date = b4fp.groupby('user_no')['report_date'].apply(list).tolist()
sess_date_diff_avg = []
for i in sess_date:
    total = 0
    if len(i) == 1: sess_date_diff_avg.append(total)
    else:
        for j in range(len(i)):
            try:
                first = datetime.datetime.strptime(str(i[j]),"%Y%m%d")
                second = datetime.datetime.strptime(str(i[j+1]),"%Y%m%d")
                total += ((second-first).days)
            except:
                break
        sess_date_diff_avg.append((total/float(len(i))))

#sum,mean,var
ssb4fp = b4fp.groupby('user_no').sum()['report_value'].tolist()
smb4fp = b4fp.groupby('user_no').mean()['report_value'].tolist()
svb4fp = b4fp.groupby('user_no').var()['report_value'].tolist()
smab4fp = b4fp.groupby('user_no').max()['report_value'].tolist()
```

Merging and outputting as csv

```
session = pd.DataFrame()
session['user_no'] = sib4fp
session['no_usage_before_purch'] = sub4fp
session['usage_total_before_purch'] = ssb4fp
session['freq_usage_before_purch'] = sfb4fp
session['session_mean_before_purch'] = smb4fp
session['session_var_before_purch'] = svb4fp
session['usage_increase_before_purch'] = sinc
session['session_max_before_purch'] = smab4fp
session['sess_date_diff_avg'] = sess_date_diff_avg

conn = pd.DataFrame()
conn['user_no'] = cib4fp
conn['no_connection_before_purch'] = cub4fp
conn['connection_total_before_purch'] = csb4fp
conn['freq_connection_before_purch'] = cfb4fp
conn['connection_mean_before_purch'] = cmb4fp
conn['connection_var_before_purch'] = cvb4fp
conn['connection_increase_before_purch'] = cinc
conn['conn_date_diff_avg'] = conn_date_diff_avg

sc = pd.merge(session,conn,on="user_no")

fsc = pd.merge(backup,sc,on="user_no")
fsc = pd.merge(fsc,userinfo,on="user_no",how="left")
fsc.to_csv('/Users/Lwmformula/Downloads/Asgn3/fsc.csv',index=False)
```

Name: Chan Ka Tsun
SID: 53563363

```
###################################
########### Asgn 3 ###############
###################################

packages <- c("foreign", "party", "rpart", "data.table","caret","e1071","partykit",
              "Formula", "aod", "ggplot2", "ROCR")
lapply(packages, library, character.only = TRUE)

#df <- read.table("~/Downloads/Network&Security_$4xx/fsc.csv",header=TRUE,sep=",")
#write.dta(df,"~/Downloads/Network&Security_$4xx/fsc.dta")
df <- read.dta("~/Downloads/Network&Security_$4xx/fsc.dta")


######### Rptree ############
set.seed(5)
sub <- df[,c(3:12,14:30)]
temp <- sample(2, nrow(sub), replace=TRUE, prob=c(0.7,0.3))
train_data <- sub[temp==1,]
test_data <- sub[temp==2,]
fit.rp<-rpart(is_dolphin~., data=train_data)

pre<-predict(fit.rp, test_data, type="matrix")
predr<-prediction(pre, test_data$is_dolphin, label.ordering = NULL)
test_data$prob <- pre
tpr_fpr.perfr <- performance(predr, measure="tpr", x.measure="fpr")
plot(tpr_fpr.perfr)
abline(a=0, b= 1)
auc.perf = performance(predr, measure = "auc")
auc.perf@y.values
opt.cut = function(perf, pred){
  cut.ind = mapply(FUN=function(x, y, p){
    d = (x-0)^2 + (y-1)^2
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, pred@cutoffs)
}
print(opt.cut(tpr_fpr.perfr, predr))

png(file="~/Downloads/Network&Security_$4xx/rptree.png",width=3500,height=1000)
plot(fit.rp, main="RP Tree")
text(fit.rp, all=TRUE, use.n=TRUE)
dev.off()
######### Rptree ############

####### All ROC CURVE #######
plot(tpr_fpr.perfr, colorize = TRUE)
plot(tpr_fpr.perfc, add = TRUE, colorize = TRUE)
plot(tpr_fpr.perfl, add = TRUE, colorize = TRUE)
abline(a=0, b= 1)
####### All ROC CURVE #######
```

Name: Chan Ka Tsun
SID: 53563363

```
######### Ctree #############
set.seed(5)
sub <- df[,c(3:12,14:30)]
temp <- sample(2, nrow(sub), replace=TRUE, prob=c(0.7,0.3))
train_data <- sub[temp==1,]
test_data <- sub[temp==2,]
fit.c <- ctree(is_dolphin~., data=train_data)

pre<-predict(fit.c, test_data, type="response")
predc<-prediction(pre, test_data$is_dolphin, label.ordering = NULL)
test_data$prob <- pre
tpr_fpr.perfc <- performance(predc, measure="tpr", x.measure="fpr")
plot(tpr_fpr.perfc)
abline(a=0, b= 1)
auc.perf = performance(predc, measure = "auc")
auc.perf@y.values
opt.cut = function(perf, pred){
  cut.ind = mapply(FUN=function(x, y, p){
    d = (x-0)^2 + (y-1)^2
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, pred@cutoffs)
}
print(opt.cut(tpr_fpr.perfc, predc))

png(file="~/Downloads/Network&Security_$4xx/Ctree.png",width=3500,height=1000)
plot(fit.c, main="Conditional Inference Tree")
dev.off()
######### Ctree #############
```

```
######### Logit #############
set.seed(5)
sub <- df[,c(3:12,14:30)]
temp <- sample(2, nrow(sub), replace=TRUE, prob=c(0.7,0.3))
train_data <- sub[temp==1,]
test_data <- sub[temp==2,]
logit <- glm(is_dolphin~. , data = train_data, family = "binomial")
predict.logit<-predict.glm(logit, newdata=test_data, type="response", se.fit=FALSE)
pred<-prediction(predict.logit, test_data$is_dolphin, label.ordering = NULL)
test_data$prob <- predict.logit

# ROC curve
tpr_fpr.perfl <- performance(pred, measure="tpr", x.measure="fpr")
plot(tpr_fpr.perfl)
abline(a=0, b= 1)

# AUC
auc.perf = performance(pred, measure = "auc")
auc.perf@y.values

# Optimal Cutoff Point
opt.cut = function(perf, pred){
    cut.ind = mapply(FUN=function(x, y, p){
        d = (x-0)^2 + (y-1)^2
        ind = which(d == min(d))
        c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
            cutoff = p[[ind]])
    }, perf@x.values, perf@y.values, pred@cutoffs)
}
print(opt.cut(tpr_fpr.perfl, pred))
######### Logit #############
```