In [ ]:

```
#Loading Libraries

library(tidyverse)
library(dplyr)
library(data.table)
library(e1071)
library(caret)
library(rpart)
library(ROSE)
library(corrplot)
library(DescTools)
library(caTools)
library(class)
library(randomForest)
library(naivebayes)
library(klaR)
library(scales)
library(neuralnet)
library(GGally)
library(PRROC)
```

In [3]:

```
#Confusion Matrix visualisation Function
draw_confusion_matrix <- function(cm) {

  layout(matrix(c(1,1,2)))
  par(mar=c(2,2,2,2))
  plot(c(100, 345), c(300, 450), type = "n", xlab="", ylab=""
, xaxt='n', yaxt='n')
  title('CONFUSION MATRIX', cex.main=2)

  # create the matrix
  rect(150, 430, 240, 370, col='#3F97D0')
```

```
  text(195, 435, 'Class0', cex=1.2)
  rect(250, 430, 340, 370, col='#F7AD50')
  text(295, 435, 'Class1', cex=1.2)
text(125, 370, 'Predicted', cex=1.3, srt=90, font=2)
text(245, 450, 'Actual', cex=1.3, font=2)
rect(150, 305, 240, 365, col='#F7AD50')
rect(250, 305, 340, 365, col='#3F97D0')
text(140, 400, 'Class0', cex=1.2, srt=90)
text(140, 335, 'Class1', cex=1.2, srt=90)

# add in the cm results
res <- as.numeric(cm$table)
text(195, 400, res[1], cex=1.6, font=2, col='white')
text(195, 335, res[2], cex=1.6, font=2, col='white')
text(295, 400, res[3], cex=1.6, font=2, col='white')
text(295, 335, res[4], cex=1.6, font=2, col='white')

# add in the specifics
plot(c(100, 0), c(100, 0), type = "n", xlab="", ylab="", main
 = "DETAILS", xaxt='n', yaxt='n')
text(10, 85, names(cm$byClass[1]), cex=1.2, font=2)
text(10, 70, round(as.numeric(cm$byClass[1]), 3), cex=1.2)
text(30, 85, names(cm$byClass[2]), cex=1.2, font=2)
text(30, 70, round(as.numeric(cm$byClass[2]), 3), cex=1.2)
text(50, 85, names(cm$byClass[5]), cex=1.2, font=2)
text(50, 70, round(as.numeric(cm$byClass[5]), 3), cex=1.2)
text(70, 85, names(cm$byClass[6]), cex=1.2, font=2)
text(70, 70, round(as.numeric(cm$byClass[6]), 3), cex=1.2)
text(90, 85, names(cm$byClass[7]), cex=1.2, font=2)
text(90, 70, round(as.numeric(cm$byClass[7]), 3), cex=1.2)

# add in the accuracy information
text(30, 35, names(cm$overall[1]), cex=1.5, font=2)
text(30, 20, round(as.numeric(cm$overall[1]), 3), cex=1.4)
text(70, 35, names(cm$overall[2]), cex=1.5, font=2)
text(70, 20, round(as.numeric(cm$overall[2]), 3), cex=1.4)
}
```

In [4]:

```
#Read the Data
setwd("C:/Users/vivek/OneDrive/Desktop/Adv BA R/project/Report")
fraud_detection=fread("creditcard.csv")
names(fraud_detection)
str(fraud_detection)
table(fraud_detection$Class)
```

'Time' 'V1' 'V2' 'V3' 'V4' 'V5' 'V6' 'V7' 'V8'
'V9' 'V10' 'V11' 'V12' 'V13' 'V14' 'V15'
'V16' 'V17' 'V18' 'V19' 'V20' 'V21' 'V22'
'V23' 'V24' 'V25' 'V26' 'V27' 'V28' 'Amount'
'Class'

```
Classes 'data.table' and 'data.frame':  284807
 obs. of  31 variables:
 $ Time  : num  0 0 1 1 2 2 4 7 7 9 ...
 $ V1    : num  -1.36 1.192 -1.358 -0.966 -1.1
58 ...
 $ V2    : num  -0.0728 0.2662 -1.3402 -0.1852
 0.8777 ...
 $ V3    : num  2.536 0.166 1.773 1.793 1.549
...
 $ V4    : num  1.378 0.448 0.38 -0.863 0.403
...
 $ V5    : num  -0.3383 0.06 -0.5032 -0.0103 -
0.4072 ...
 $ V6    : num  0.4624 -0.0824 1.8005 1.2472 0
.0959 ...
 $ V7    : num  0.2396 -0.0788 0.7915 0.2376 0
.5929 ...
 $ V8    : num  0.0987 0.0851 0.2477 0.3774 -0
.2705 ...
 $ V9    : num  0.364 -0.255 -1.515 -1.387 0.8
18 ...
 $ V10   : num  0.0908 -0.167 0.2076 -0.055 0.
```

```
7531 ...
 $ V11   : num  -0.552 1.613 0.625 -0.226 -0.8
23 ...
 $ V12   : num  -0.6178 1.0652 0.0661 0.1782 0
.5382 ...
 $ V13   : num  -0.991 0.489 0.717 0.508 1.346
 ...
 $ V14   : num  -0.311 -0.144 -0.166 -0.288 -1
.12 ...
 $ V15   : num  1.468 0.636 2.346 -0.631 0.175
 ...
 $ V16   : num  -0.47 0.464 -2.89 -1.06 -0.451
 ...
 $ V17   : num  0.208 -0.115 1.11 -0.684 -0.23
7 ...
 $ V18   : num  0.0258 -0.1834 -0.1214 1.9658
-0.0382 ...
 $ V19   : num  0.404 -0.146 -2.262 -1.233 0.8
03 ...
 $ V20   : num  0.2514 -0.0691 0.525 -0.208 0.
4085 ...
 $ V21   : num  -0.01831 -0.22578 0.248 -0.108
3 -0.00943 ...
 $ V22   : num  0.27784 -0.63867 0.77168 0.005
27 0.79828 ...
 $ V23   : num  -0.11 0.101 0.909 -0.19 -0.137
 ...
 $ V24   : num  0.0669 -0.3398 -0.6893 -1.1756
 0.1413 ...
 $ V25   : num  0.129 0.167 -0.328 0.647 -0.20
6 ...
 $ V26   : num  -0.189 0.126 -0.139 -0.222 0.5
02 ...
 $ V27   : num  0.13356 -0.00898 -0.05535 0.06
272 0.21942 ...
 $ V28   : num  -0.0211 0.0147 -0.0598 0.0615
0.2152 ...
```

```
 $ Amount: num  149.62 2.69 378.66 123.5 69.99
 ...
 $ Class : int  0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

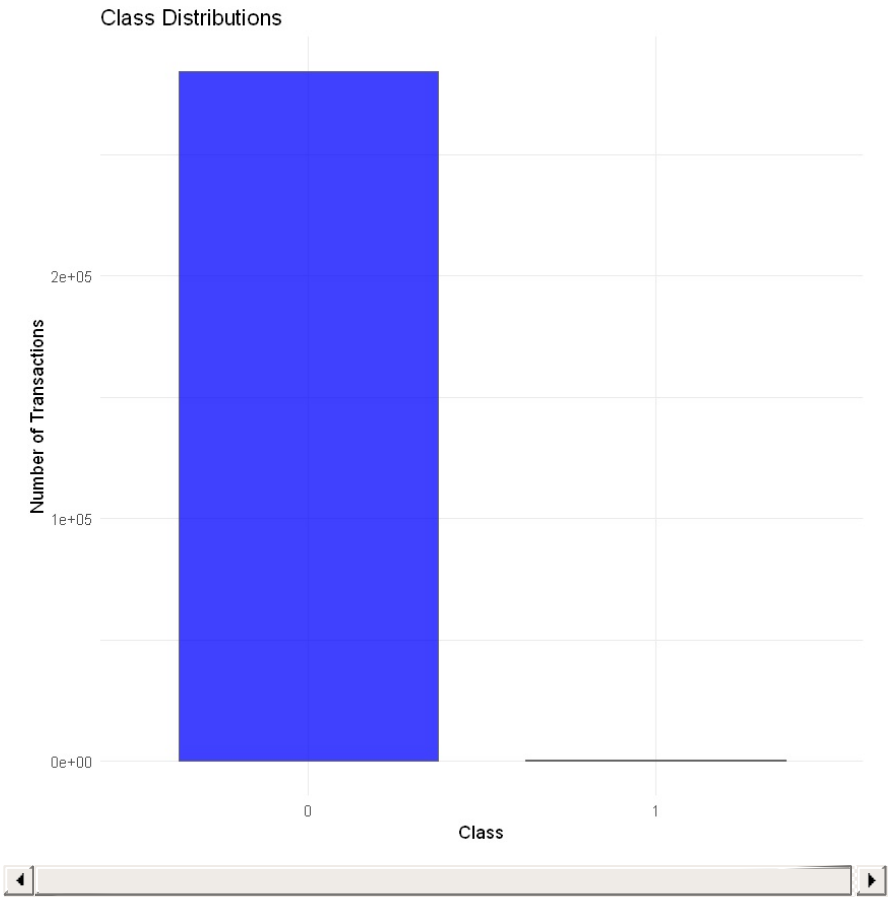
```
     0      1
284315    492
```

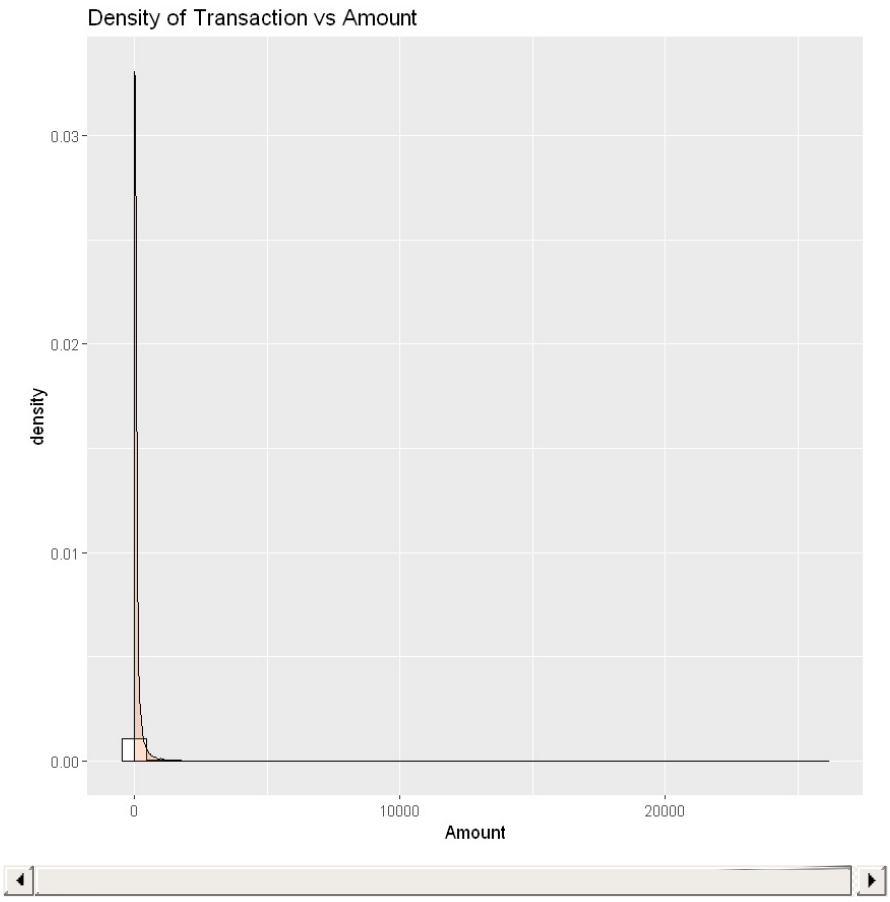
Class Distribution

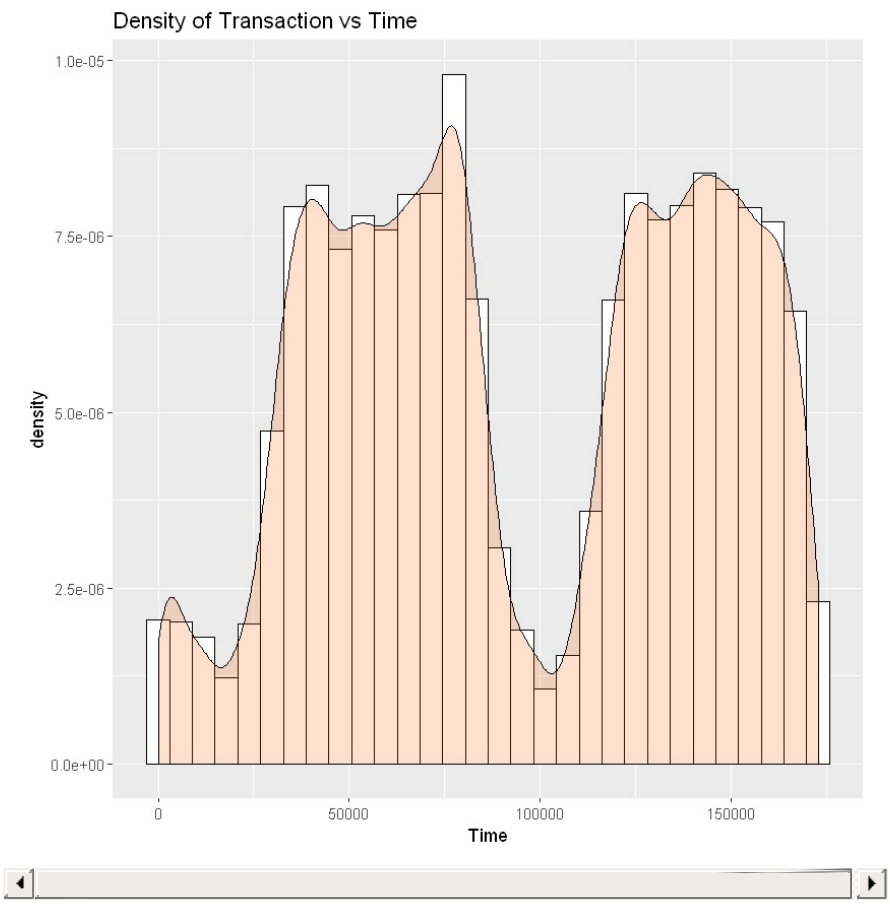
In [5]:

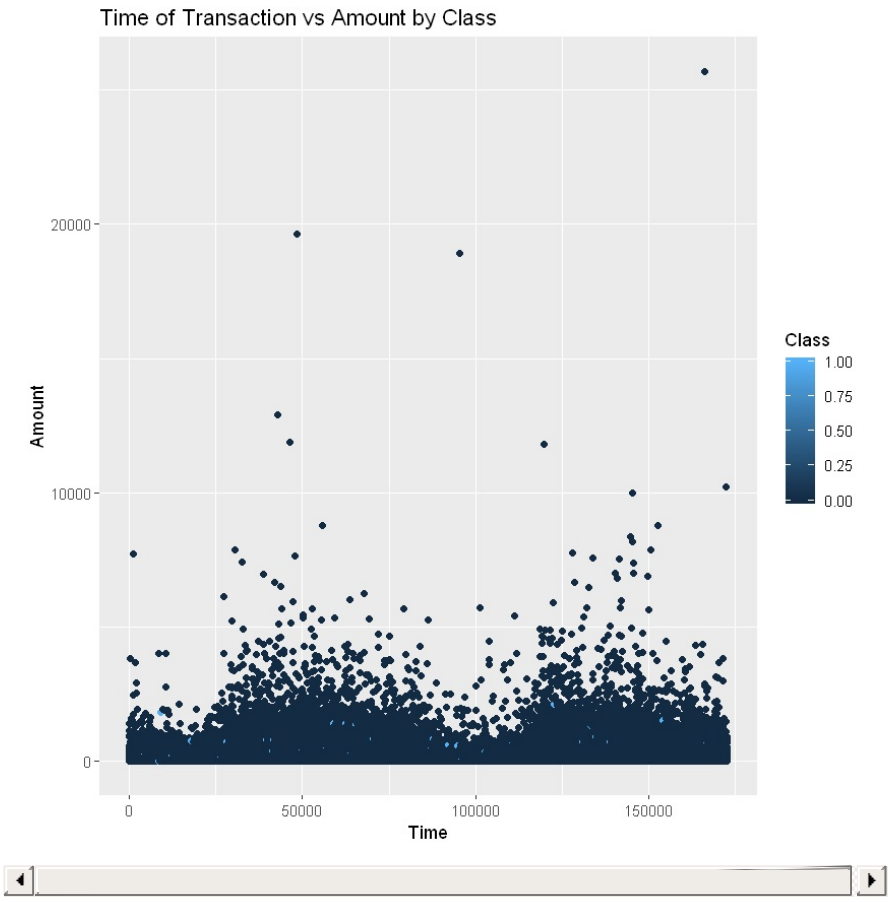
```
x34=fraud_detection[,-1]
y34=fraud_detection[,31]

#visualising the class

y34 %>%
  mutate(max_class = max(table(Class))) %>%
  ggplot(aes(x=factor(Class)))+
  geom_bar(stat="count", width=0.75, fill="blue", color = "gr
ey40", alpha = .75)+
  xlab("Class") + ylab("Number of Transactions") +
  ggtitle("Class Distributions") +
  theme_minimal()

#Amount Density

ggplot(fraud_detection, aes(x=Amount)) +
  geom_histogram(aes(y=..density..),stat="bin",bins="30", col
our="black", fill="white")+
  geom_density(alpha=.2, fill="#FF6611") +
  ggtitle("Density of Transaction vs Amount")

#Time Density

ggplot(fraud_detection, aes(x=Time)) +
  geom_histogram(aes(y=..density..),stat="bin",bins="30", col
our="black", fill="white")+
  geom_density(alpha=.2, fill="#FF6611") +
  ggtitle("Density of Transaction vs Time")
```

```r
#time amount and class
ggplot(filter(fraud_detection, Class %in% c("0", "1")),
       aes(x=Time,
           y=Amount,
           color=Class))+
  geom_point() +
  ggtitle("Time of Transaction vs Amount by Class")
```




Density of Transaction vs Amount
0.03
0.02
0.01
0.00
density
0
10000
20000
Amount


Density of Transaction vs Time
1.0e-05
7.5e-06
5.0e-06
2.5e-06
0.0e+00
density
0
50000
100000
150000
Time



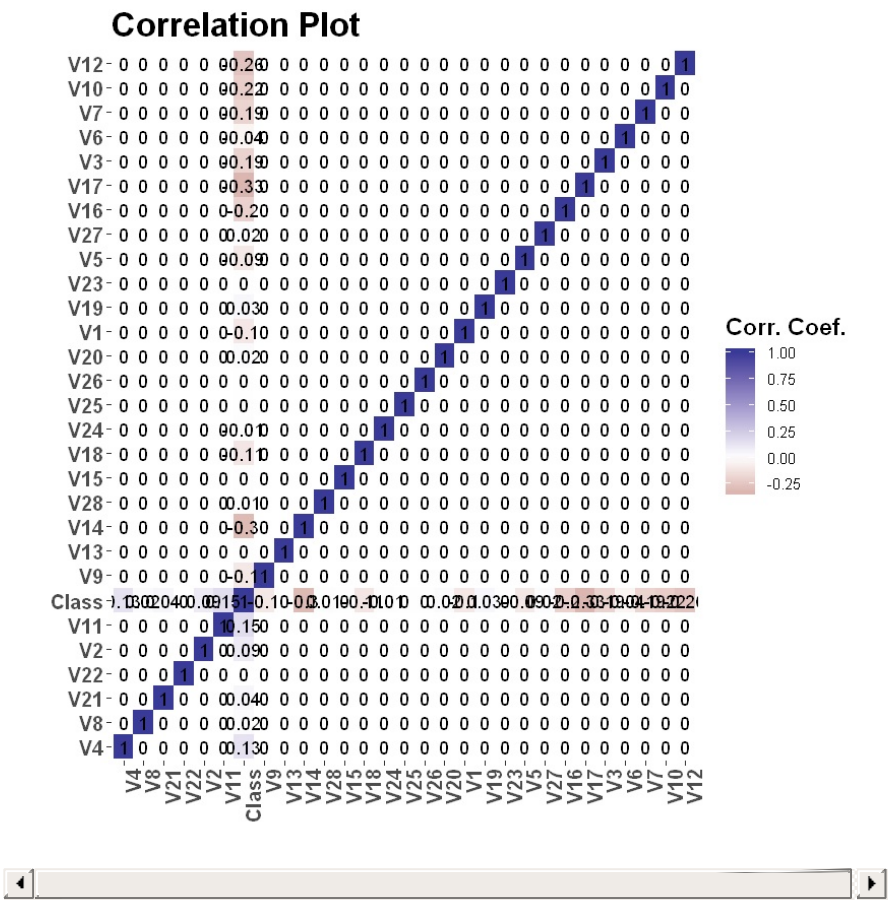
In [8]:

```
#correlation

fraud_detection_c=fraud_detection
fraud_detection_c=fraud_detection_c[,c(-1,-30)]
#fraud_detection_c$Class=as.numeric(fraud_detection_c$Class)
fraud_detection_cp=as.matrix(fraud_detection_c)
class(fraud_detection_cp)
corr_mat=cor(fraud_detection_cp,method="s")
corr_mat=cor(fraud_detection_cp)
library(scales)
ord=hclust(1-as.dist(corr_mat))$order
co=melt(corr_mat[ord,ord])
ggplot(co, aes(Var1, Var2)) +
  geom_tile(aes(fill = value)) +
  geom_text(aes(fill = co$value, label = round(co$value, 2)))
```

```
 +
  scale_fill_gradient2(low = muted("darkred"),
                       mid = "white",
                       high = muted("midnightblue"),
                       midpoint = 0) +
  theme(panel.grid.major.x=element_blank(),
        panel.grid.minor.x=element_blank(),
        panel.grid.major.y=element_blank(),
        panel.grid.minor.y=element_blank(),
        panel.background=element_rect(fill="white"),
        axis.text.x = element_text(angle=90, hjust = 1,vjust=
1,size = 12,face = "bold"),
        plot.title = element_text(size=20,face="bold"),
        axis.text.y = element_text(size = 12,face = "bold"))
+
  ggtitle("Correlation Plot") +
  theme(legend.title=element_text(face="bold", size=14)) +
  scale_x_discrete(name="") +
  scale_y_discrete(name="") +
  labs(fill="Corr. Coef.")
```

'matrix'

```
Warning message:
"Ignoring unknown aesthetics: fill"
```



In [6]:

```
#Scaling the data

fraud_detection$scaled_time=RobScale(fraud_detection$Time,cen
ter=TRUE,scale = TRUE)
fraud_detection$scaled_amount=RobScale(fraud_detection$Amount,
center=TRUE,scale = TRUE)

fraud_detection_1=fraud_detection
fraud_detection_1$Time=NULL
fraud_detection_1$Amount=NULL
fraud_detection_dt=fraud_detection
fraud_detection_1$Class <- as.factor(fraud_detection_1$Class)
x=fraud_detection[,-1]
y=fraud_detection[,31]
```

In [7]:

```
#Training and Testing Split

#test and train split

trainindex <- createDataPartition(fraud_detection_1$Class, p=
0.8, list= FALSE)
fd_train <- fraud_detection_1[trainindex,]
fd_test <- fraud_detection_1[-trainindex, ]

#for dt test and train

trainindex_dt <- createDataPartition(fraud_detection_dt$Class
, p=0.8, list= FALSE)
fd_train_dt <- fraud_detection_dt[trainindex_dt,]
fd_test_dt<- fraud_detection_dt[-trainindex_dt, ]
```

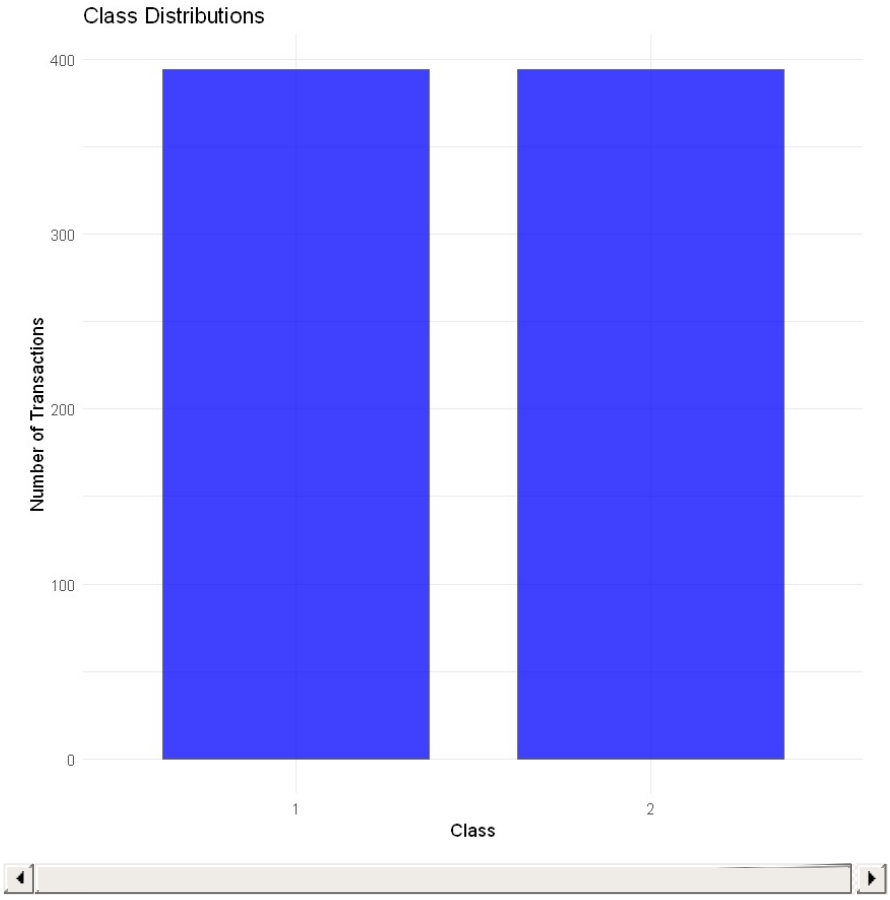
In [8]:

```
#Value for N in Sampling and Correlation

#value for n in sampling

x=as.data.frame(table(fd_train$Class))
x1=x%>%filter(Var1==0)
x11=x1$Freq
x2=x%>%filter(Var1==1)
x22=x2$Freq

#value for n in sampling fpr dt

x_dt=as.data.frame(table(fd_train_dt$Class))
x1_dt=x_dt%>%filter(Var1==0)
x11_dt=x1_dt$Freq
x2_dt=x_dt%>%filter(Var1==1)
x22_dt=x2_dt$Freq
```

In [9]:

```
#Undersampling

fd_balanced_under <- ovun.sample(Class ~ ., data = fd_train,
method = "under",
                                 N = (2*x22), seed = 1)$data
table(fd_balanced_under$Class)

y111=fd_balanced_under[,c(29,30)]
y111$Class=as.numeric(y111$Class)

#visualising the class
ggplot(y111,aes(x=factor(Class)))+
  geom_bar(stat="count", width=0.75, fill="blue", color = "gr
ey40", alpha = .75)+
  xlab("Class") + ylab("Number of Transactions") +
  ggtitle("Class Distributions") +
  theme_minimal()
```

```
  0   1
394 394
```

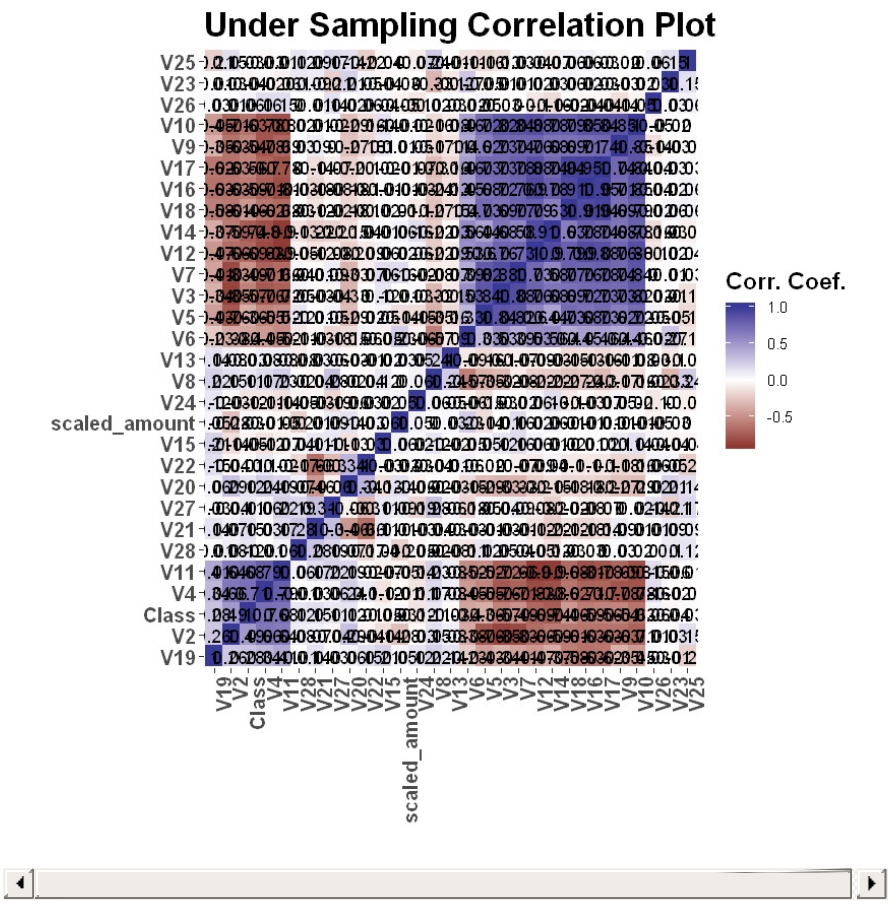


In [15]:

```
#Undersapling correlation

fraud_detection_c=fd_balanced_under
fraud_detection_c=fraud_detection_c[,c(-1,-30)]
fraud_detection_c$Class=as.numeric(fraud_detection_c$Class)
fraud_detection_cp=as.matrix(fraud_detection_c)
corr_mat=cor(fraud_detection_cp,method="s")
corr_mat=cor(fraud_detection_cp)
ord=hclust(1-as.dist(corr_mat))$order
co=melt(corr_mat[ord,ord])
ggplot(co, aes(Var1, Var2)) +
  geom_tile(aes(fill = value)) +
  geom_text(aes(fill = co$value, label = round(co$value, 2)))
 +
  scale_fill_gradient2(low = muted("darkred"),
```

```
                      mid = "white",
                      high = muted("midnightblue"),
                      midpoint = 0) +
  theme(panel.grid.major.x=element_blank(),
        panel.grid.minor.x=element_blank(),
        panel.grid.major.y=element_blank(),
        panel.grid.minor.y=element_blank(),
        panel.background=element_rect(fill="white"),
        axis.text.x = element_text(angle=90, hjust = 1,vjust=
1,size = 12,face = "bold"),
        plot.title = element_text(size=20,face="bold"),
        axis.text.y = element_text(size = 12,face = "bold"))
+
  ggtitle("Under Sampling Correlation Plot") +
  theme(legend.title=element_text(face="bold", size=14)) +
  scale_x_discrete(name="") +
  scale_y_discrete(name="") +
  labs(fill="Corr. Coef.")
```

```
Warning message:
"Ignoring unknown aesthetics: fill"
```



In [18]:

```
#Algorithms
#Logistic Regression

classifier_us = glm(formula = Class ~ .,family = binomial,dat
a =fd_balanced_under)
pred_log_us = predict(classifier_us, type = 'response', newda
ta = fd_test)
pred_log_us_1 = ifelse(pred_log_us > 0.5, 1, 0)
```

```
Warning message:
"glm.fit: algorithm did not converge"Warning m
essage:
"glm.fit: fitted probabilities numerically 0 o
r 1 occurred"
```

In [19]:

```
#Logistic Confusion matrix and ROC

cm1=confusionMatrix(table(pred_log_us_1,fd_test$Class))
cm1
draw_confusion_matrix(cm1)
x1=roc.curve(fd_test$Class, pred_log_us_1,curve = TRUE)
x1
plot(x1)
a1=(pr.curve(fd_test$Class, pred_log_us_1,curve = TRUE))
a1
plot(a1)
```

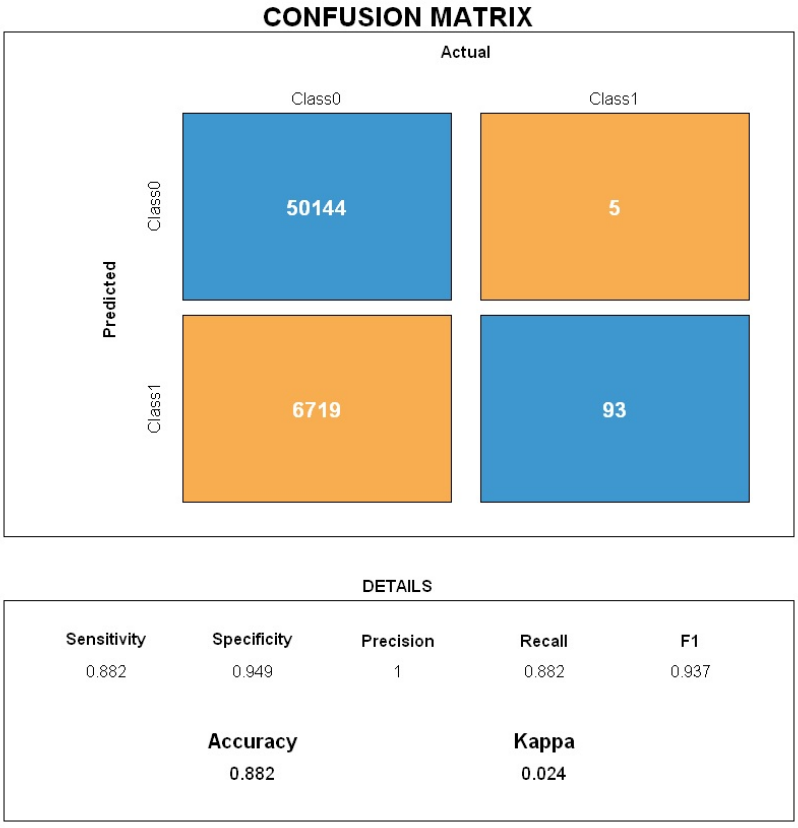
```
Confusion Matrix and Statistics


pred_log_us_1     0     1
            0 50144     5
            1  6719    93

               Accuracy : 0.882
                 95% CI : (0.8793, 0.8846)
    No Information Rate : 0.9983
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0236
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.88184
            Specificity : 0.94898
         Pos Pred Value : 0.99990
         Neg Pred Value : 0.01365
             Prevalence : 0.99828
         Detection Rate : 0.88032
   Detection Prevalence : 0.88041
      Balanced Accuracy : 0.91541
```

'Positive' Class : 0

ROC curve

Area under curve:
0.9403076

Curve for scores from 0 to 2
(can be plotted with plot(x))

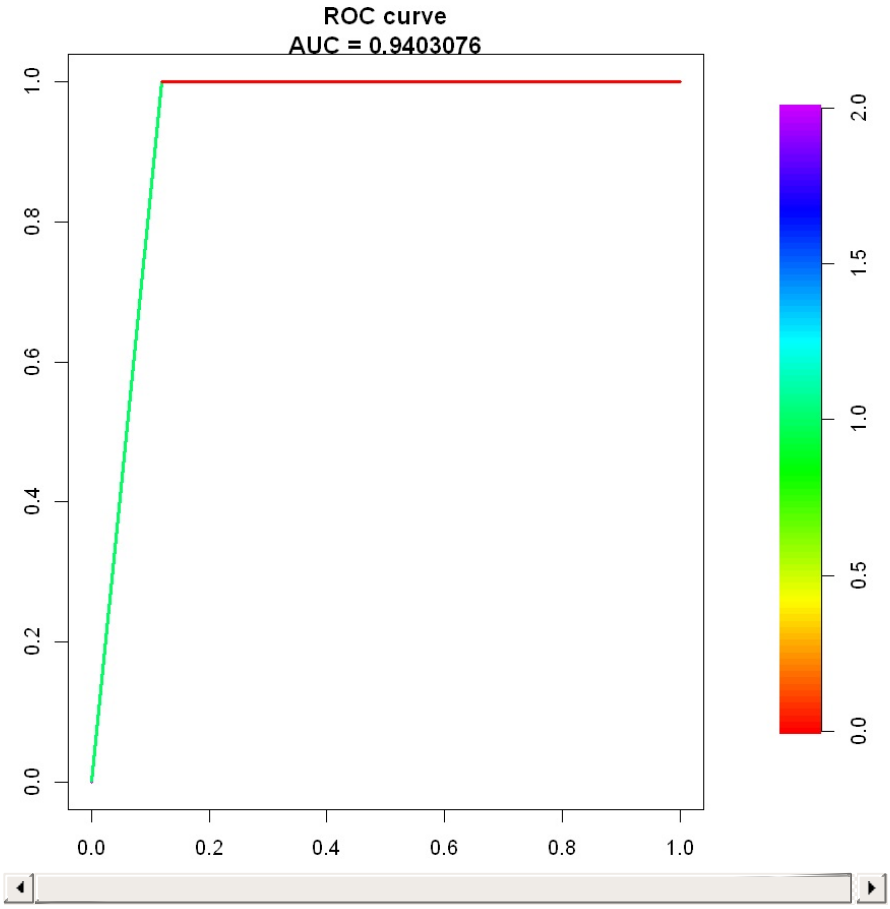


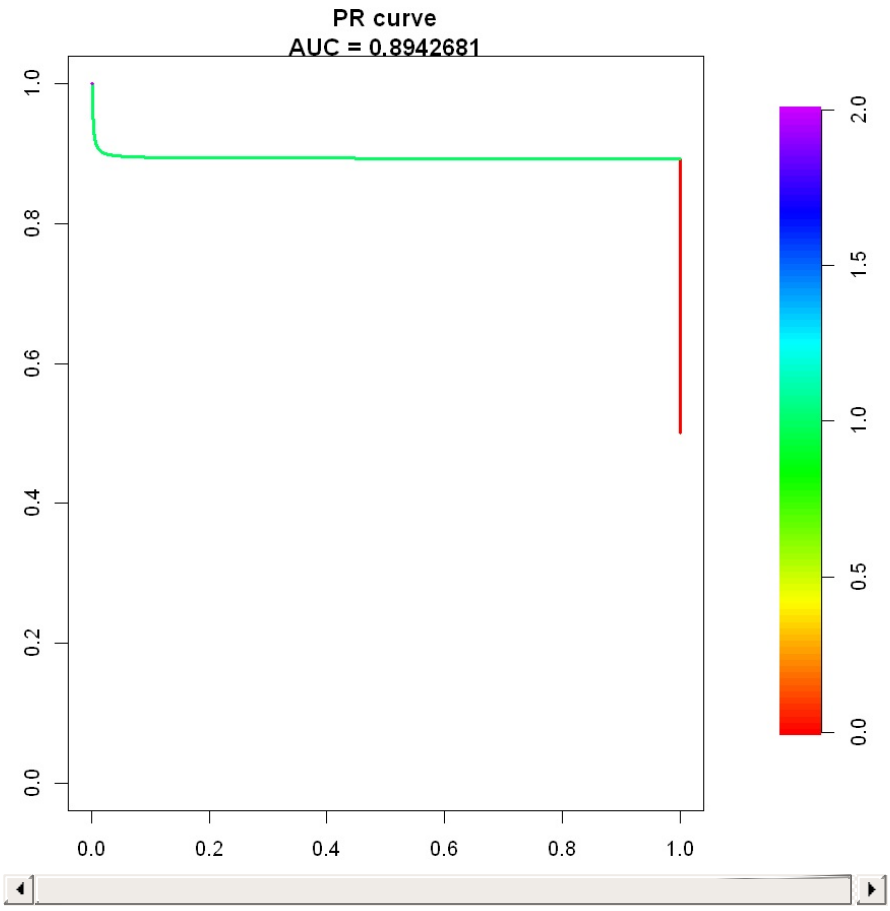
Precision-recall curve

Area under curve (Integral):
0.8942681

Area under curve (Davis & Goadrich):

```
0.8942681

Curve for scores from  0  to  2
( can be plotted with plot(x) )
```





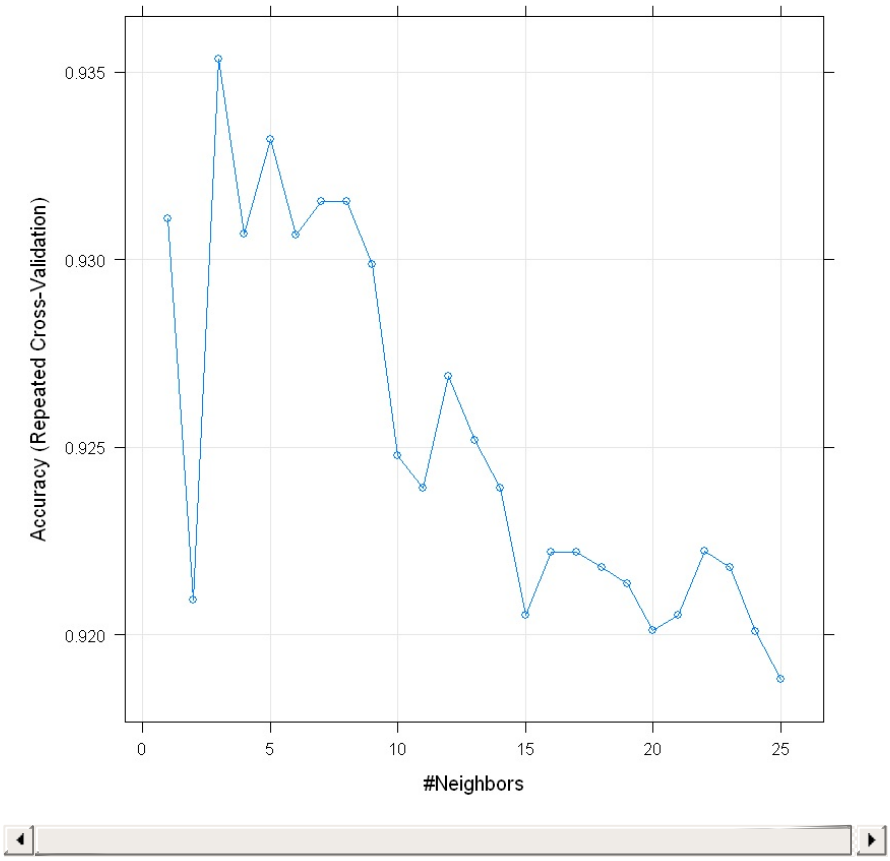
In [20]:

```
#Knn
#Hyperparameter tunning

knn_us <- train(Class~., data=fd_balanced_under, method='knn'
,
  tuneGrid=expand.grid(.k=1:25),metric='Accuracy',
  trControl=trainControl(method='repeatedcv', number=10,repea
ts=3))
knn_us
knn_us_df=as.data.frame(knn_us$results)
knn_us_optimal=max(knn_us_df$k)
plot(knn_us)
```

```
k-Nearest Neighbors
```

```
788 samples
 30 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated
 3 times)
Summary of sample sizes: 710, 709, 709, 708, 7
09, 710, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.9310854  0.8621531
   2  0.9209152  0.8418221
   3  0.9353271  0.8706516
   4  0.9306854  0.8613564
   5  0.9332118  0.8664279
   6  0.9306639  0.8613396
   7  0.9315455  0.8631015
   8  0.9315455  0.8631074
   9  0.9298576  0.8597411
  10  0.9247779  0.8495859
  11  0.9239069  0.8478334
  12  0.9268929  0.8538103
  13  0.9251889  0.8503932
  14  0.9239124  0.8478357
  15  0.9205259  0.8410741
  16  0.9222030  0.8444289
  17  0.9222084  0.8444376
  18  0.9217972  0.8436212
  19  0.9213644  0.8427453
  20  0.9201039  0.8402358
  21  0.9205150  0.8410571
  22  0.9222188  0.8444595
  23  0.9217809  0.8435797
  24  0.9200822  0.8401822
  25  0.9188272  0.8376767
```

```
Accuracy was used to select the optimal model
using the largest value.
The final value used for the model was k = 3.
```



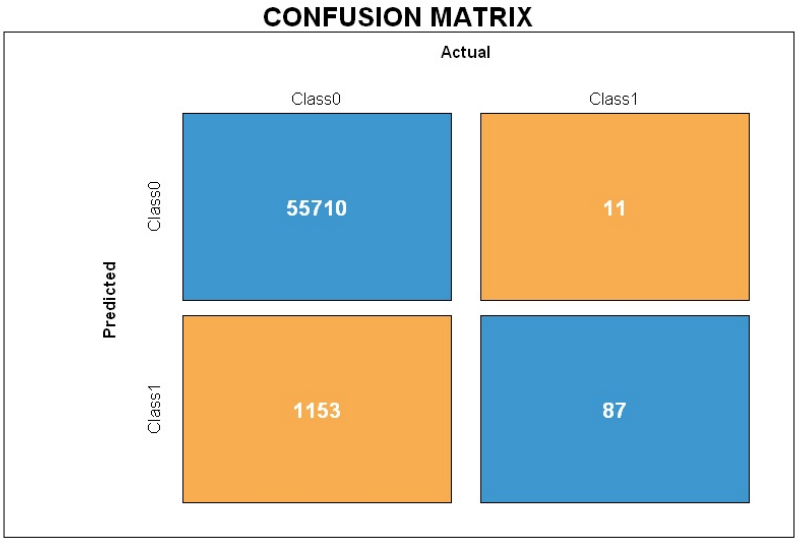
In [21]:

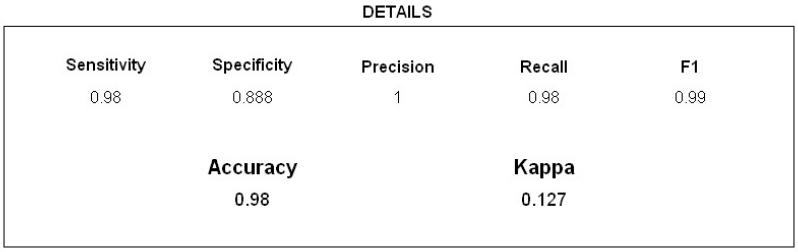
```
#Train, Confusion matrix and ROC
pred_knn_us = knn(train = fd_balanced_under[,-29],test = fd_t
est[,-29],
              cl = fd_balanced_under[,29],
              k = knn_us_optimal,
              prob = TRUE)

cm2=confusionMatrix(table(pred_knn_us,fd_test$Class))
cm2
draw_confusion_matrix(cm2)
x1=roc.curve(fd_test$Class, pred_knn_us,curve = TRUE)
```

```
x1
plot(x1)
a2=(pr.curve(fd_test$Class, pred_knn_us,curve = TRUE))
a2
plot(a2)
```

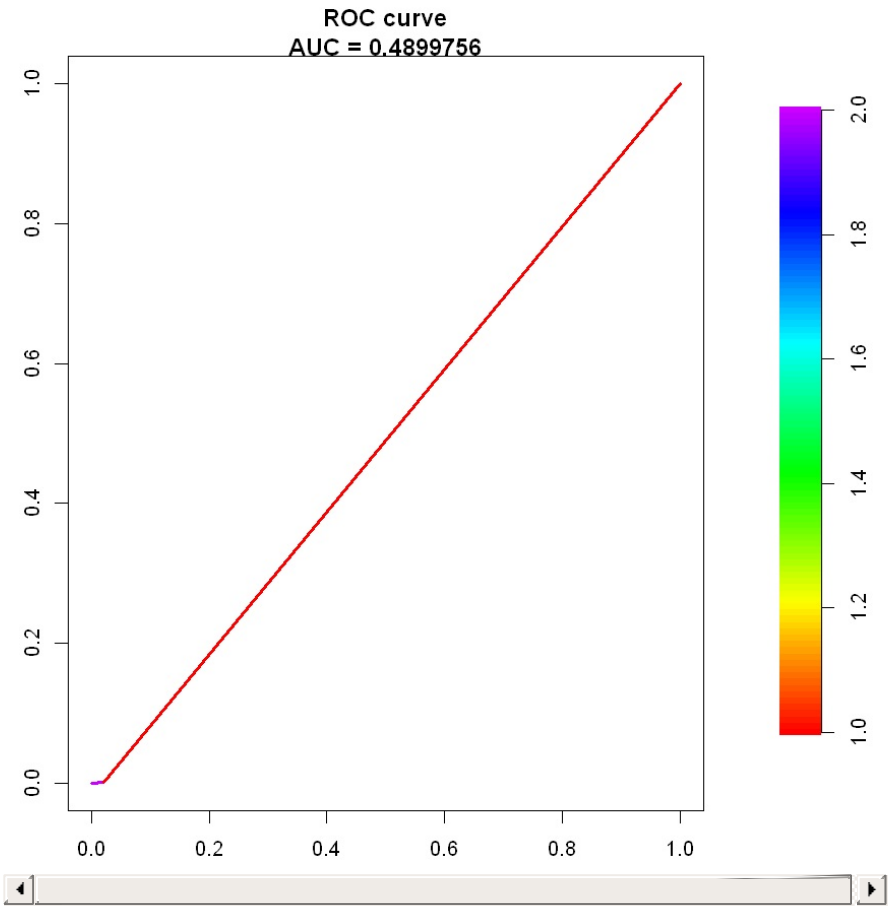
```
Confusion Matrix and Statistics


pred_knn_us     0     1
          0 55710    11
          1  1153    87

               Accuracy : 0.9796
                 95% CI : (0.9784, 0.9807)
    No Information Rate : 0.9983
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1273
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.97972
            Specificity : 0.88776
         Pos Pred Value : 0.99980
         Neg Pred Value : 0.07016
             Prevalence : 0.99828
         Detection Rate : 0.97804
   Detection Prevalence : 0.97823
      Balanced Accuracy : 0.93374

       'Positive' Class : 0


  ROC curve

    Area under curve:
     0.4899756
```

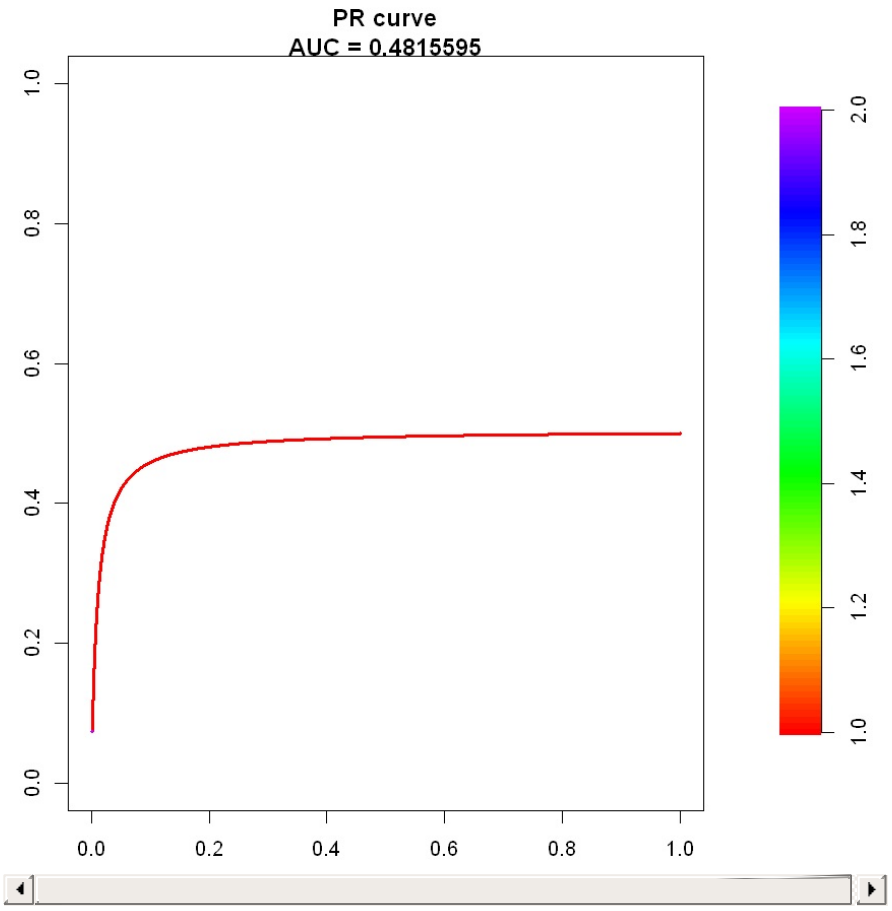
```
Curve for scores from  1  to  2
( can be plotted with plot(x) )
```






```
Precision-recall curve

  Area under curve (Integral):
   0.4815595

  Area under curve (Davis & Goadrich):
   0.4815595

  Curve for scores from  1  to  2
  ( can be plotted with plot(x) )
```


ROC curve
AUC = 0.4899756
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0



In [23]:

```
#Naive Bayes

#Hyperparameter tunning
options(warn=-1)
nb_us = train(x = fd_balanced_under[-29],
              y = fd_balanced_under$Class,method = "nb",
              trControl = trainControl(method='repeatedcv', n
umber=10,repeats=3),
              tuneGrid = expand.grid(usekernel = c(TRUE, FALS
E),fL = 0:5,adjust = seq(0, 5, by = 1)))

nb_us
```

```
Naive Bayes
```

```
788 samples
 30 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated
 3 times)
Summary of sample sizes: 710, 709, 708, 708, 7
10, 710, ...
Resampling results across tuning parameters:

  usekernel  fL  adjust  Accuracy   Kappa
  FALSE      0   0       0.9095507  0.8191050
  FALSE      0   1       0.9095507  0.8191050
  FALSE      0   2       0.9095507  0.8191050
  FALSE      0   3       0.9095507  0.8191050
  FALSE      0   4       0.9095507  0.8191050
  FALSE      0   5       0.9095507  0.8191050
  FALSE      1   0       0.9095507  0.8191050
  FALSE      1   1       0.9095507  0.8191050
  FALSE      1   2       0.9095507  0.8191050
  FALSE      1   3       0.9095507  0.8191050
  FALSE      1   4       0.9095507  0.8191050
  FALSE      1   5       0.9095507  0.8191050
  FALSE      2   0       0.9095507  0.8191050
  FALSE      2   1       0.9095507  0.8191050
  FALSE      2   2       0.9095507  0.8191050
  FALSE      2   3       0.9095507  0.8191050
  FALSE      2   4       0.9095507  0.8191050
  FALSE      2   5       0.9095507  0.8191050
  FALSE      3   0       0.9095507  0.8191050
  FALSE      3   1       0.9095507  0.8191050
  FALSE      3   2       0.9095507  0.8191050
  FALSE      3   3       0.9095507  0.8191050
  FALSE      3   4       0.9095507  0.8191050
  FALSE      3   5       0.9095507  0.8191050
  FALSE      4   0       0.9095507  0.8191050
```

FALSE	4	1	0.9095507	0.8191050
FALSE	4	2	0.9095507	0.8191050
FALSE	4	3	0.9095507	0.8191050
FALSE	4	4	0.9095507	0.8191050
FALSE	4	5	0.9095507	0.8191050
FALSE	5	0	0.9095507	0.8191050
FALSE	5	1	0.9095507	0.8191050
FALSE	5	2	0.9095507	0.8191050
FALSE	5	3	0.9095507	0.8191050
FALSE	5	4	0.9095507	0.8191050
FALSE	5	5	0.9095507	0.8191050
TRUE	0	0	NaN	NaN
TRUE	0	1	0.9052882	0.8105772
TRUE	0	2	0.9010524	0.8020965
TRUE	0	3	0.8972386	0.7944552
TRUE	0	4	0.8908926	0.7817677
TRUE	0	5	0.8791038	0.7582076
TRUE	1	0	NaN	NaN
TRUE	1	1	0.9052882	0.8105772
TRUE	1	2	0.9010524	0.8020965
TRUE	1	3	0.8972386	0.7944552
TRUE	1	4	0.8908926	0.7817677
TRUE	1	5	0.8791038	0.7582076
TRUE	2	0	NaN	NaN
TRUE	2	1	0.9052882	0.8105772
TRUE	2	2	0.9010524	0.8020965
TRUE	2	3	0.8972386	0.7944552
TRUE	2	4	0.8908926	0.7817677
TRUE	2	5	0.8791038	0.7582076
TRUE	3	0	NaN	NaN
TRUE	3	1	0.9052882	0.8105772
TRUE	3	2	0.9010524	0.8020965
TRUE	3	3	0.8972386	0.7944552
TRUE	3	4	0.8908926	0.7817677
TRUE	3	5	0.8791038	0.7582076
TRUE	4	0	NaN	NaN
TRUE	4	1	0.9052882	0.8105772

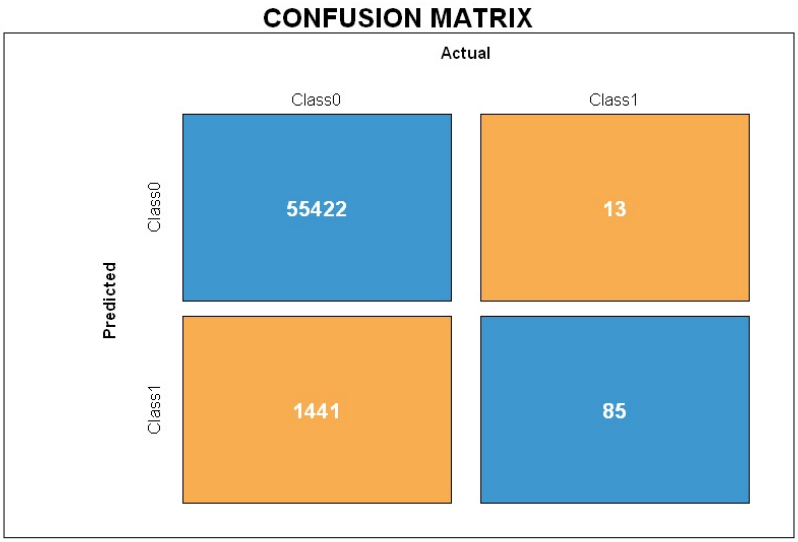
```
   TRUE      4   2         0.9010524  0.8020965
   TRUE      4   3         0.8972386  0.7944552
   TRUE      4   4         0.8908926  0.7817677
   TRUE      4   5         0.8791038  0.7582076
   TRUE      5   0               NaN        NaN
   TRUE      5   1         0.9052882  0.8105772
   TRUE      5   2         0.9010524  0.8020965
   TRUE      5   3         0.8972386  0.7944552
   TRUE      5   4         0.8908926  0.7817677
   TRUE      5   5         0.8791038  0.7582076

Accuracy was used to select the optimal model
using the largest value.
The final values used for the model were fL =
0, usekernel = FALSE and adjust
 = 0.
```

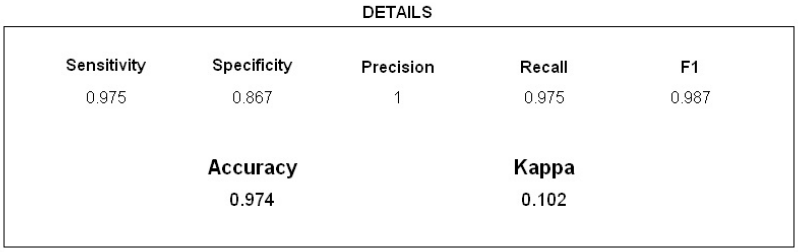
In [24]:

```
#Confusion matrix and ROC

pred_nb_us = predict(nb_us, newdata = fd_test)
cm3=confusionMatrix(table(pred_nb_us,fd_test$Class))
cm3
draw_confusion_matrix(cm3)
x3=roc.curve(fd_test$Class, pred_nb_us,curve = TRUE)
x3
plot(x3)

#recall

a3=(pr.curve(fd_test$Class, pred_nb_us,curve = TRUE))
a3
plot(a3)
```

```
Confusion Matrix and Statistics
```

```
pred_nb_us     0     1
         0 55422    13
         1  1441    85

               Accuracy : 0.9745
                 95% CI : (0.9731, 0.9758)
    No Information Rate : 0.9983
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1018
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9747
            Specificity : 0.8673
         Pos Pred Value : 0.9998
         Neg Pred Value : 0.0557
             Prevalence : 0.9983
         Detection Rate : 0.9730
   Detection Prevalence : 0.9732
      Balanced Accuracy : 0.9210

       'Positive' Class : 0


  ROC curve

    Area under curve:
     0.4874651

    Curve for scores from  1  to  2
    ( can be plotted with plot(x) )
```



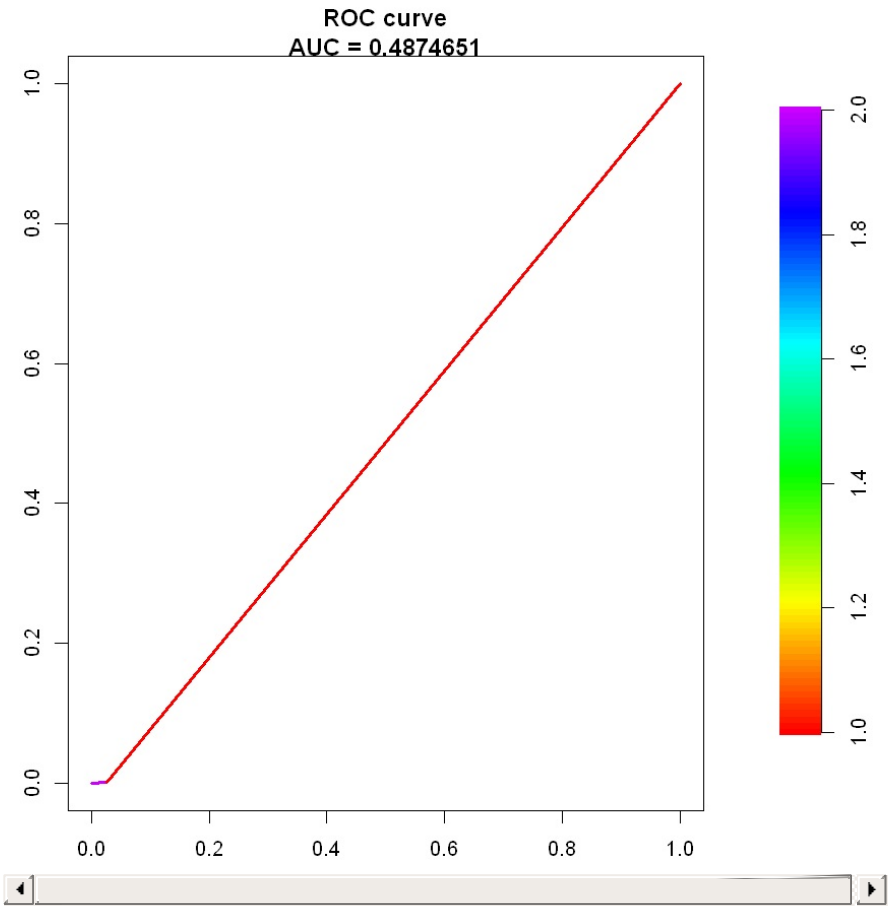


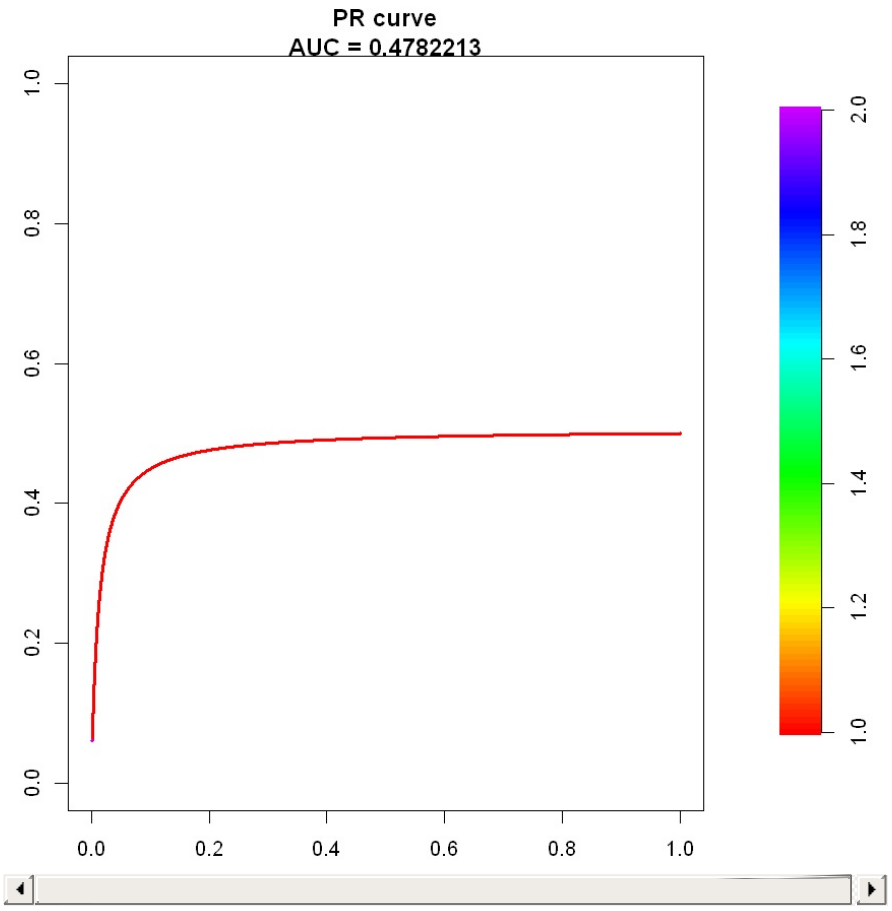
Precision-recall curve

Area under curve (Integral):
0.4782213

Area under curve (Davis & Goadrich):
0.4782213

Curve for scores from 1 to 2
(can be plotted with plot(x))


ROC curve
AUC = 0.4874651
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0



In [25]:

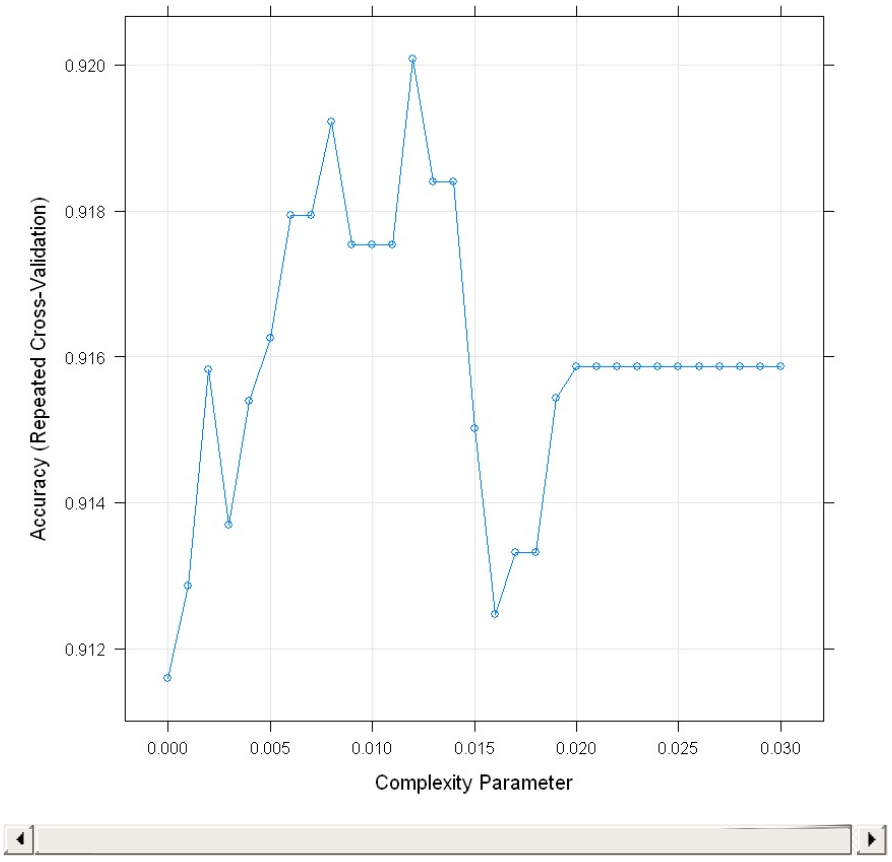
```
#Decision Tree
#Hyperparameter tunning
options(warn=-1)
dt_us = train(Class~., data=fd_balanced_under, method='rpart'
,
              tuneGrid=expand.grid(.cp=seq(0.00,0.03,0.001)),
metric='Accuracy',
              trControl=trainControl(method='repeatedcv', num
ber=10,repeats=3))
dt_us
plot(dt_us)
```

CART

788 samples

```
 30 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated
 3 times)
Summary of sample sizes: 709, 710, 710, 709, 7
08, 709, ...
Resampling results across tuning parameters:

  cp     Accuracy   Kappa
  0.000  0.9115973  0.8230818
  0.001  0.9128631  0.8256277
  0.002  0.9158224  0.8315552
  0.003  0.9136858  0.8272714
  0.004  0.9153952  0.8306902
  0.005  0.9162553  0.8324068
  0.006  0.9179432  0.8357964
  0.007  0.9179432  0.8357964
  0.008  0.9192252  0.8383605
  0.009  0.9175373  0.8349942
  0.010  0.9175373  0.8349942
  0.011  0.9175373  0.8349942
  0.012  0.9200747  0.8400708
  0.013  0.9183973  0.8367161
  0.014  0.9183973  0.8367161
  0.015  0.9150212  0.8299823
  0.016  0.9124732  0.8248853
  0.017  0.9133225  0.8265838
  0.018  0.9133225  0.8265838
  0.019  0.9154376  0.8308064
  0.020  0.9158650  0.8316611
  0.021  0.9158650  0.8316611
  0.022  0.9158650  0.8316611
  0.023  0.9158650  0.8316611
  0.024  0.9158650  0.8316611
  0.025  0.9158650  0.8316611
```

```
0.026  0.9158650  0.8316611
0.027  0.9158650  0.8316611
0.028  0.9158650  0.8316611
0.029  0.9158650  0.8316611
0.030  0.9158650  0.8316611

Accuracy was used to select the optimal model
using the largest value.
The final value used for the model was cp = 0.
012.
```



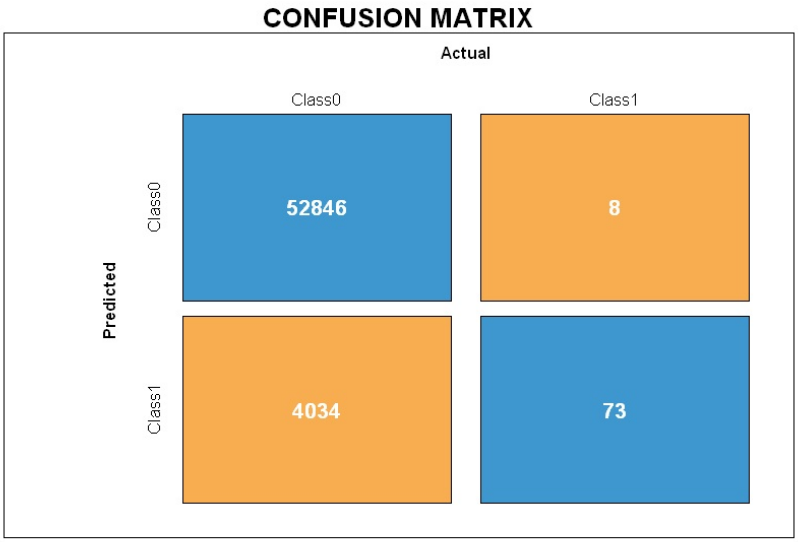
In [26]:

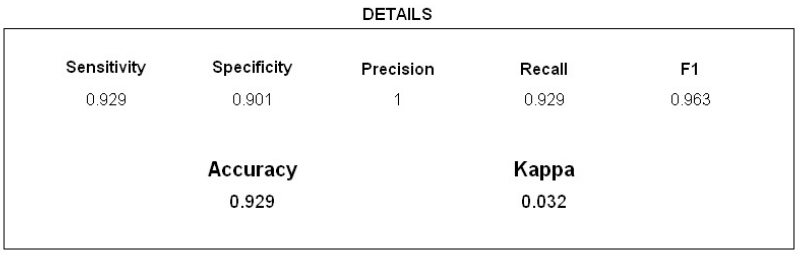
```
#Confusion matrix and ROC

fd_balanced_under_dt <- ovun.sample(Class ~ ., data = fd_trai
n_dt, method = "under",
                                    N = (2*x22_dt), seed = 1)$da
```

```
ta
table(fd_balanced_under_dt$Class)

tree_us = rpart(Class ~ ., data =fd_balanced_under_dt,
                control=rpart.control(cp=0,minbucket= 8,minsp
lit = 100))
prune_us <- prune(tree_us, cp = 0.012)
pred_tree_us <- predict(prune_us, newdata = fd_test_dt)
pred_tree_us_1 = ifelse(pred_tree_us > 0.5, 1, 0)

cm4=confusionMatrix(table(pred_tree_us_1,fd_test_dt$Class))
cm4
draw_confusion_matrix(cm4)
x4=roc.curve(fd_test_dt$Class, pred_tree_us_1,curve=TRUE)
x4
plot(x4)
#recall

a4=(pr.curve(fd_test$Class, pred_tree_us_1,curve = TRUE))
a4
plot(a4)
```

```
  0   1
411 411

Confusion Matrix and Statistics


pred_tree_us_1     0     1
             0 52846     8
             1  4034    73

               Accuracy : 0.929
                 95% CI : (0.9269, 0.9311)
    No Information Rate : 0.9986
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0322
```

```
Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.92908
            Specificity : 0.90123
         Pos Pred Value : 0.99985
         Neg Pred Value : 0.01777
             Prevalence : 0.99858
         Detection Rate : 0.92776
   Detection Prevalence : 0.92790
      Balanced Accuracy : 0.91516

       'Positive' Class : 0


 ROC curve

   Area under curve:
    0.46466

   Curve for scores from  0  to  1
   ( can be plotted with plot(x) )
```



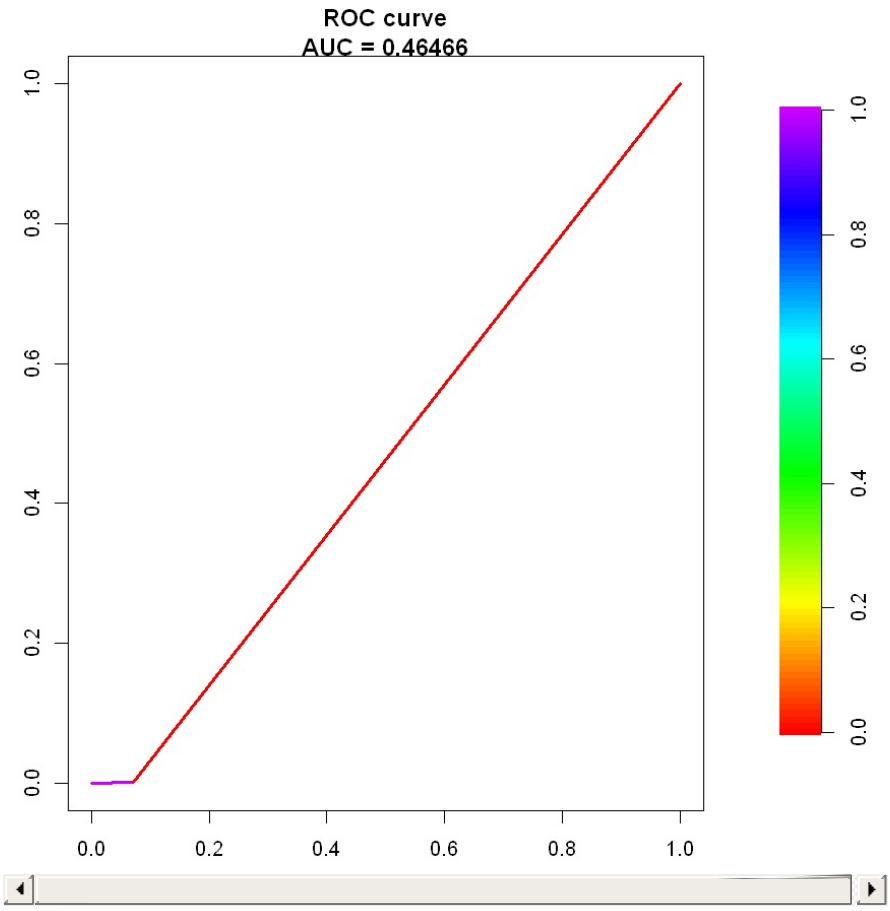


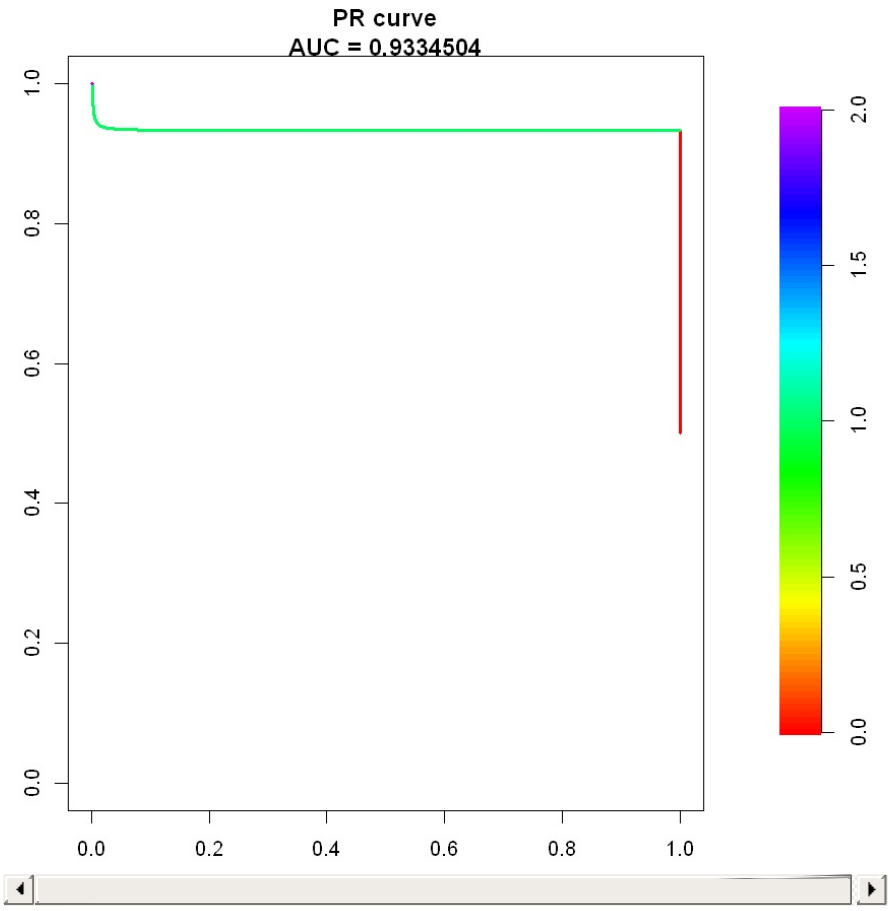
Precision-recall curve

Area under curve (Integral):
0.9334504

Area under curve (Davis & Goadrich):
0.9334504

Curve for scores from 0 to 2
(can be plotted with plot(x))


ROC curve
AUC = 0.46466
0.0
0.2
0.4
0.6
0.8
1.0



In [27]:

```
#Ramdom Forest
#Hyperparameter tunning

options(warn=-1)
rf_us = train(Class~., data=fd_balanced_under, method='rf',
               tuneGrid=expand.grid(.mtry=c(1:15)),
              metric='Accuracy',trControl=trainControl(method
='repeatedcv', number=10,repeats=2))
rf_us
```

```
Random Forest

788 samples
 30 predictor
  2 classes: '0', '1'
```

```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated
 2 times)
Summary of sample sizes: 709, 708, 710, 709, 7
09, 710, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   1    0.9295188  0.8589423
   2    0.9327240  0.8653696
   3    0.9359047  0.8717395
   4    0.9384445  0.8768144
   5    0.9358966  0.8717207
   6    0.9390853  0.8781070
   7    0.9365456  0.8730249
   8    0.9371624  0.8742641
   9    0.9384524  0.8768463
  10    0.9384362  0.8768183
  11    0.9403430  0.8806348
  12    0.9378193  0.8755864
  13    0.9377870  0.8755271
  14    0.9390610  0.8780732
  15    0.9365454  0.8730419

Accuracy was used to select the optimal model
using the largest value.
The final value used for the model was mtry =
11.
```

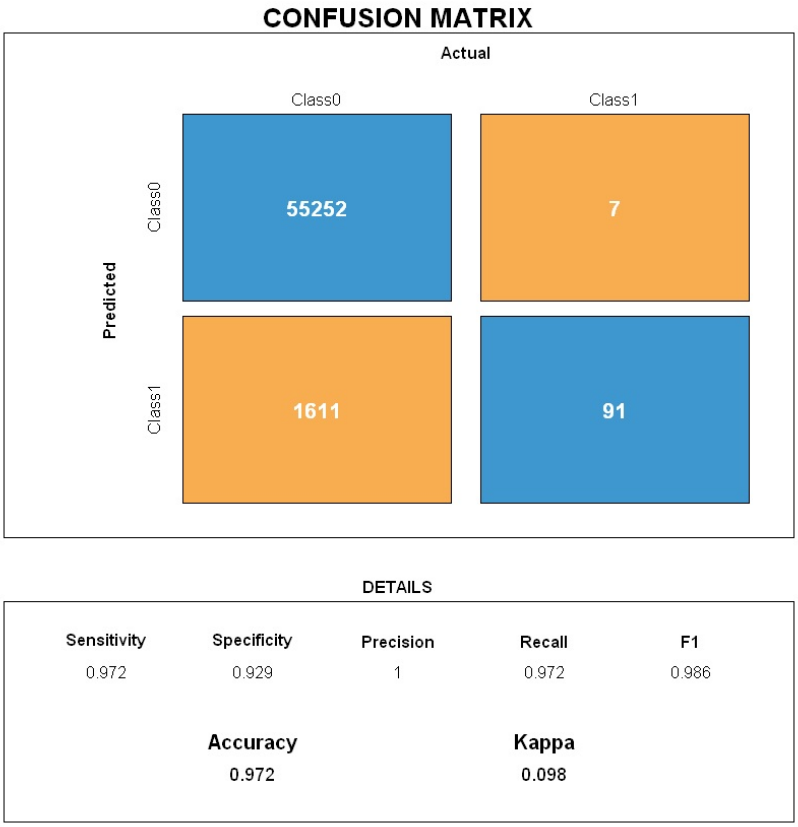
In [29]:

```
#Confusion matrix and ROC

randomforest = randomForest(x = fd_balanced_under[-29],
                            y = fd_balanced_under$Class,
                            ntree=1500,mtry =11)
```

```
pred_rf_us = predict(randomforest, newdata = fd_test)
cm5=confusionMatrix(table(pred_rf_us,fd_test$Class))
cm5
draw_confusion_matrix(cm5)
x5=roc.curve(fd_test$Class, pred_rf_us,curve = TRUE)
x5
plot(x5)


#recall

a5=(pr.curve(fd_test$Class, pred_rf_us,curve = TRUE))
a5
plot(a5)
```

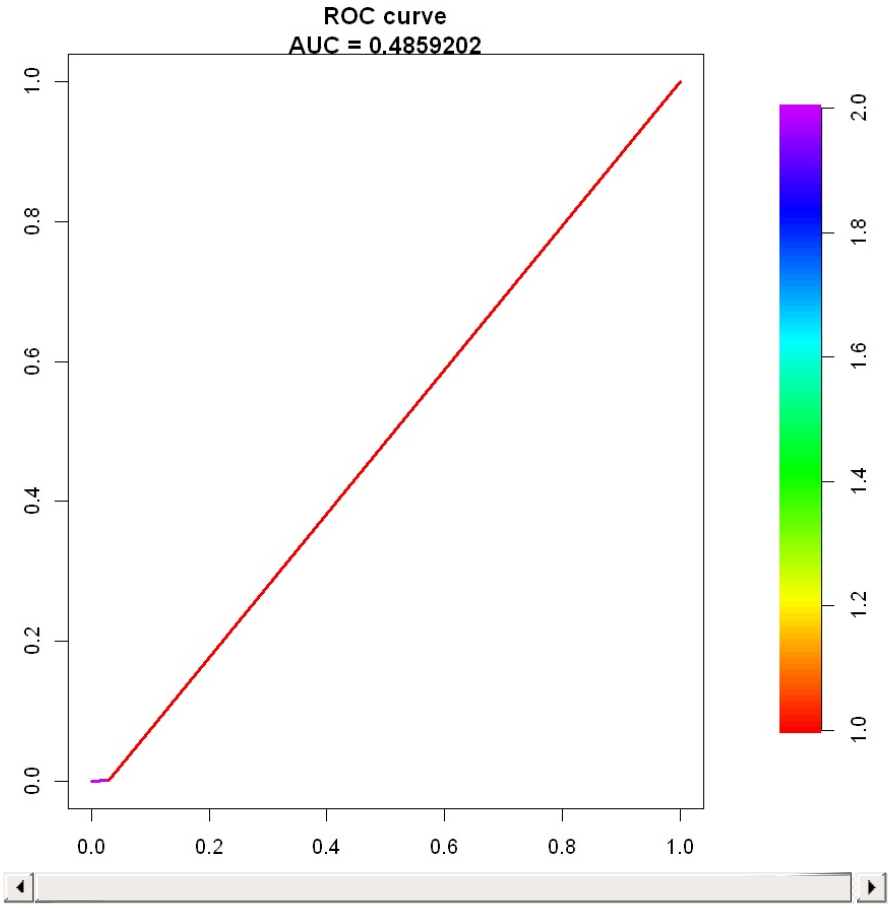
```
Confusion Matrix and Statistics


pred_rf_us     0     1
         0 55252     7
         1  1611    91

               Accuracy : 0.9716
                 95% CI : (0.9702, 0.9729)
    No Information Rate : 0.9983
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0982
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.97167
            Specificity : 0.92857
         Pos Pred Value : 0.99987
         Neg Pred Value : 0.05347
             Prevalence : 0.99828
         Detection Rate : 0.97000
```

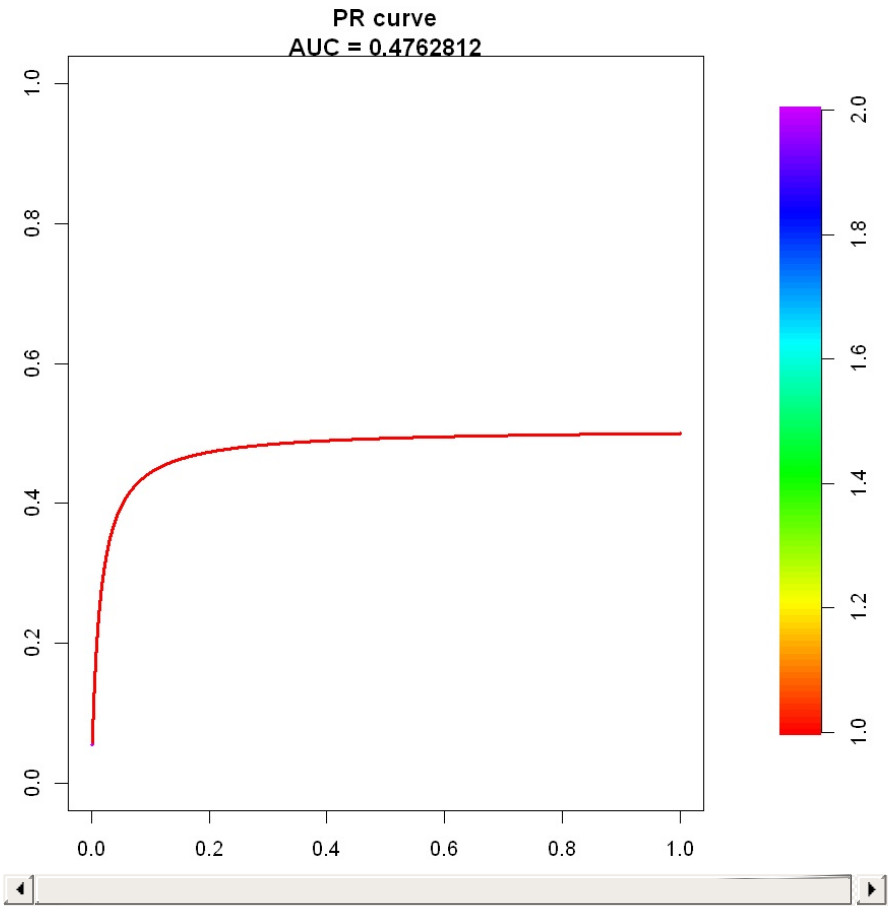
```
 Detection Prevalence : 0.97012
    Balanced Accuracy : 0.95012

     'Positive' Class : 0


ROC curve

  Area under curve:
   0.4859202

  Curve for scores from  1  to  2
  ( can be plotted with plot(x) )
```




```
Precision-recall curve

  Area under curve (Integral):
   0.4762812
```

```
Area under curve (Davis & Goadrich):
 0.4762812

Curve for scores from  1  to  2
( can be plotted with plot(x) )
```





In [30]:

```
#Support Vector Machine
#Hyperparameter tuning
options(warn=-1)
svm_tn_us <- train(Class ~., data = fd_balanced_under,
                   method = "svmPoly",
                   trControl=trainControl(method = "repeatedcv",

                                          number = 10, repeats =
 1),
                   preProcess = c("center", "scale"),
                   tuneGrid = expand.grid(.degree = c(2:5),.scal
e = c(0.1,1,10),
                        .C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75
,1, 1.5, 2,5)),
                   tuneLength = 10)
```

```
svm_tn_us
plot(svm_tn_us)
```

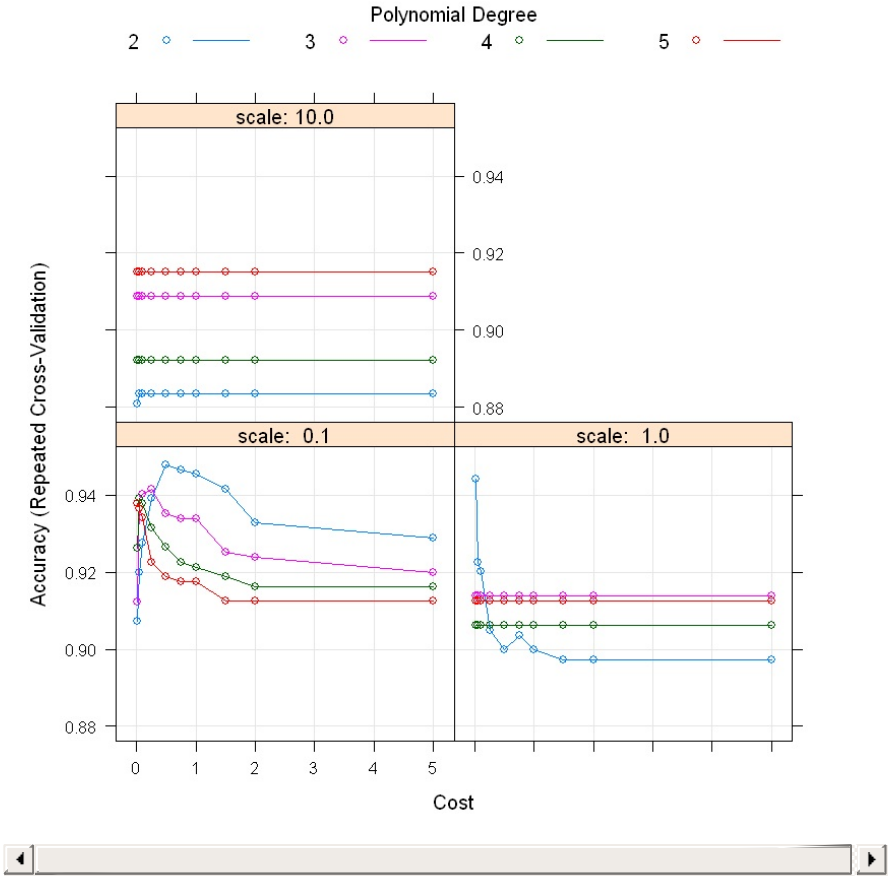
```
Support Vector Machines with Polynomial Kernel


788 samples
 30 predictor
  2 classes: '0', '1'

Pre-processing: centered (30), scaled (30)
Resampling: Cross-Validated (10 fold, repeated
 1 times)
Summary of sample sizes: 709, 709, 710, 709, 7
09, 709, ...
Resampling results across tuning parameters:

  degree  scale  C     Accuracy   Kappa
  2        0.1   0.00       NaN        NaN
  2        0.1   0.01  0.9073028  0.8144828
  2        0.1   0.05  0.9200260  0.8399626
  2        0.1   0.10  0.9276209  0.8551713
  2        0.1   0.25  0.9390782  0.8781068
  2        0.1   0.50  0.9479714  0.8958912
  2        0.1   0.75  0.9467056  0.8933519
  2        0.1   1.00  0.9454398  0.8908190
  2        0.1   1.50  0.9416261  0.8831973
  2        0.1   2.00  0.9327491  0.8654344
  2        0.1   5.00  0.9289192  0.8577685
  2        1.0   0.00       NaN        NaN
  2        1.0   0.01  0.9441740  0.8882926
  2        1.0   0.05  0.9226225  0.8451520
  2        1.0   0.10  0.9200747  0.8400697
  2        1.0   0.25  0.9048523  0.8096648
  2        1.0   0.50  0.8997728  0.7995150
  2        1.0   0.75  0.9035540  0.8070434
  2        1.0   1.00  0.8997566  0.7994221
```

2	1.0	1.50	0.8972087	0.7943154
2	1.0	2.00	0.8972087	0.7943154
2	1.0	5.00	0.8972087	0.7943154
2	10.0	0.00	NaN	NaN
2	10.0	0.01	0.8807855	0.7614620
2	10.0	0.05	0.8833171	0.7665245
2	10.0	0.10	0.8833171	0.7665245
2	10.0	0.25	0.8833171	0.7665245
2	10.0	0.50	0.8833171	0.7665245
2	10.0	0.75	0.8833171	0.7665245
2	10.0	1.00	0.8833171	0.7665245
2	10.0	1.50	0.8833171	0.7665245
2	10.0	2.00	0.8833171	0.7665245
2	10.0	5.00	0.8833171	0.7665245
3	0.1	0.00	NaN	NaN
3	0.1	0.01	0.9123986	0.8246834
3	0.1	0.05	0.9390782	0.8781197
3	0.1	0.10	0.9403603	0.8806904
3	0.1	0.25	0.9416586	0.8832776
3	0.1	0.50	0.9352970	0.8705265
3	0.1	0.75	0.9339825	0.8679162
3	0.1	1.00	0.9339662	0.8678866
3	0.1	1.50	0.9251217	0.8502170
3	0.1	2.00	0.9238559	0.8476906
3	0.1	5.00	0.9200422	0.8400478
3	1.0	0.00	NaN	NaN
3	1.0	0.01	0.9137455	0.8274199
3	1.0	0.05	0.9137455	0.8274346
3	1.0	0.10	0.9137455	0.8274346
3	1.0	0.25	0.9137455	0.8274346
3	1.0	0.50	0.9137455	0.8274346
3	1.0	0.75	0.9137455	0.8274346
3	1.0	1.00	0.9137455	0.8274346
3	1.0	1.50	0.9137455	0.8274346
3	1.0	2.00	0.9137455	0.8274346
3	1.0	5.00	0.9137455	0.8274346
3	10.0	0.00	NaN	NaN

3	10.0	0.01	0.9086822	0.8173291
3	10.0	0.05	0.9086822	0.8173291
3	10.0	0.10	0.9086822	0.8173291
3	10.0	0.25	0.9086822	0.8173291
3	10.0	0.50	0.9086822	0.8173291
3	10.0	0.75	0.9086822	0.8173291
3	10.0	1.00	0.9086822	0.8173291
3	10.0	1.50	0.9086822	0.8173291
3	10.0	2.00	0.9086822	0.8173291
3	10.0	5.00	0.9086822	0.8173291
4	0.1	0.00	NaN	NaN
4	0.1	0.01	0.9263551	0.8526466
4	0.1	0.05	0.9391107	0.8781741
4	0.1	0.10	0.9378611	0.8756531
4	0.1	0.25	0.9314671	0.8628638
4	0.1	0.50	0.9264038	0.8527486
4	0.1	0.75	0.9226225	0.8451942
4	0.1	1.00	0.9213405	0.8426577
4	0.1	1.50	0.9188251	0.8376054
4	0.1	2.00	0.9162610	0.8324805
4	0.1	5.00	0.9162772	0.8324987
4	1.0	0.00	NaN	NaN
4	1.0	0.01	0.9061506	0.8122617
4	1.0	0.05	0.9061506	0.8122617
4	1.0	0.10	0.9061506	0.8122617
4	1.0	0.25	0.9061506	0.8122617
4	1.0	0.50	0.9061506	0.8122617
4	1.0	0.75	0.9061506	0.8122617
4	1.0	1.00	0.9061506	0.8122617
4	1.0	1.50	0.9061506	0.8122617
4	1.0	2.00	0.9061506	0.8122617
4	1.0	5.00	0.9061506	0.8122617
4	10.0	0.00	NaN	NaN
4	10.0	0.01	0.8921616	0.7842898
4	10.0	0.05	0.8921616	0.7842898
4	10.0	0.10	0.8921616	0.7842898
4	10.0	0.25	0.8921616	0.7842898

4	10.0	0.50	0.8921616	0.7842898
4	10.0	0.75	0.8921616	0.7842898
4	10.0	1.00	0.8921616	0.7842898
4	10.0	1.50	0.8921616	0.7842898
4	10.0	2.00	0.8921616	0.7842898
4	10.0	5.00	0.8921616	0.7842898
5	0.1	0.00	NaN	NaN
5	0.1	0.01	0.9378124	0.8755950
5	0.1	0.05	0.9365790	0.8731052
5	0.1	0.10	0.9340474	0.8680427
5	0.1	0.25	0.9226550	0.8452615
5	0.1	0.50	0.9188251	0.8376297
5	0.1	0.75	0.9175592	0.8350758
5	0.1	1.00	0.9175755	0.8351119
5	0.1	1.50	0.9124797	0.8248920
5	0.1	2.00	0.9124797	0.8248920
5	0.1	5.00	0.9124797	0.8248920
5	1.0	0.00	NaN	NaN
5	1.0	0.01	0.9124473	0.8248489
5	1.0	0.05	0.9124473	0.8248489
5	1.0	0.10	0.9124473	0.8248489
5	1.0	0.25	0.9124473	0.8248489
5	1.0	0.50	0.9124473	0.8248489
5	1.0	0.75	0.9124473	0.8248489
5	1.0	1.00	0.9124473	0.8248489
5	1.0	1.50	0.9124473	0.8248489
5	1.0	2.00	0.9124473	0.8248489
5	1.0	5.00	0.9124473	0.8248489
5	10.0	0.00	NaN	NaN
5	10.0	0.01	0.9149951	0.8299426
5	10.0	0.05	0.9149951	0.8299426
5	10.0	0.10	0.9149951	0.8299426
5	10.0	0.25	0.9149951	0.8299426
5	10.0	0.50	0.9149951	0.8299426
5	10.0	0.75	0.9149951	0.8299426
5	10.0	1.00	0.9149951	0.8299426
5	10.0	1.50	0.9149951	0.8299426

```
  5        10.0   2.00  0.9149951  0.8299426
  5        10.0   5.00  0.9149951  0.8299426

Accuracy was used to select the optimal model
using the largest value.
The final values used for the model were degre
e = 2, scale = 0.1 and C = 0.5.
```



In [33]:

```
#SVM Tuning
options(warn=-1)
tune_out = tune.svm(x = fd_balanced_under[,-29], y = fd_balan
ced_under[,29],
           type = "C-classification", kernel = "polynomial",
degree = 2,
           cost = 0.5, gamma = c(0.1,0.5,1,10), coef0 = c(0.1
,1,10))
```

```
cost=tune_out$best.parameters$cost
cost
gamma=tune_out$best.parameters$gamma
gamma
coef0=tune_out$best.parameters$coef0
coef0
```

0.5

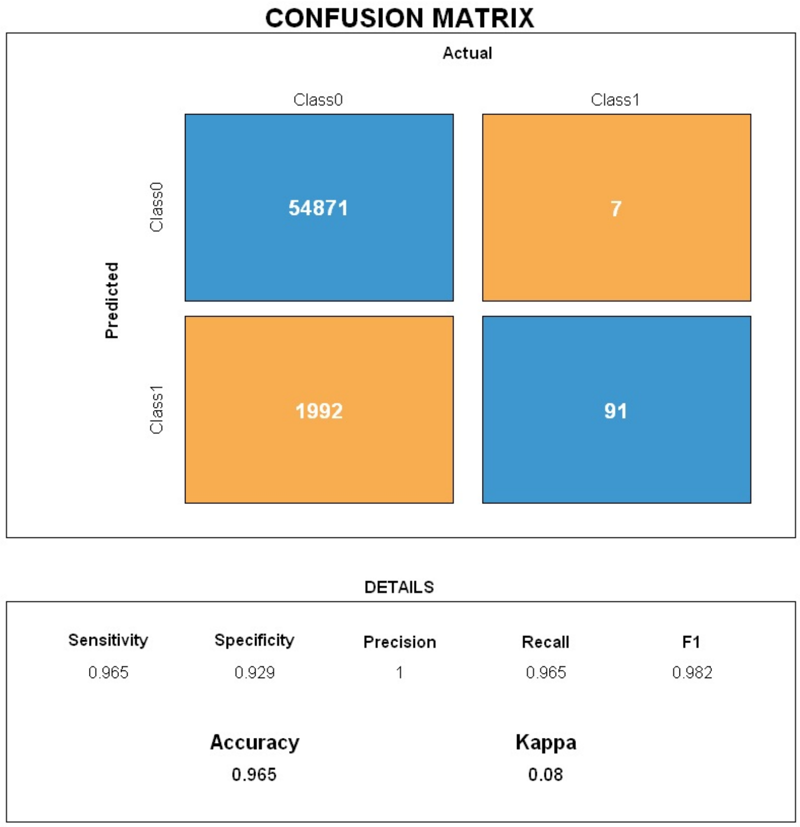
0.1

10

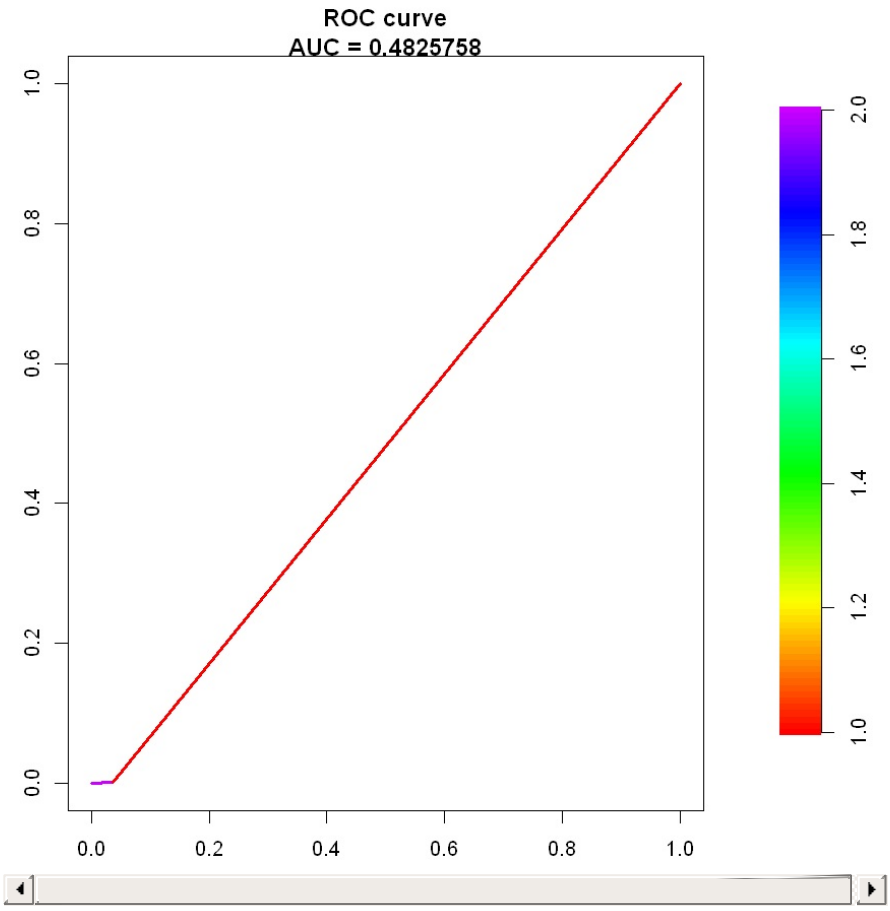
In [34]:

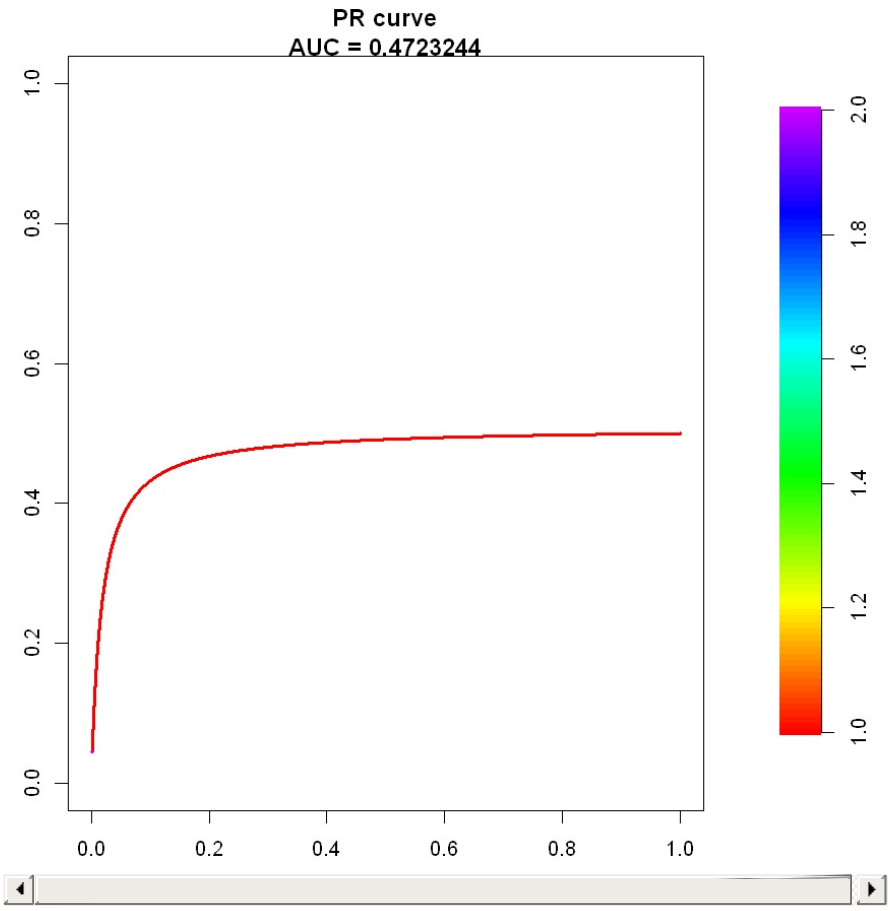
```
#Confusion matrix and ROC


svm_us <- svm(Class~ ., data = fd_balanced_under, type = "C-classification",
                 kernel = "polynomial", degree = 2,scale=0.1,
                 cost = tune_out$best.parameters$cost,
                 gamma = tune_out$best.parameters$gamma,
                 coef0 = tune_out$best.parameters$coef0)
pred_svm_us = predict(svm_us, newdata = fd_test)
cm6=confusionMatrix(table(pred_svm_us,fd_test$Class))
cm6
draw_confusion_matrix(cm6)
x6=roc.curve(fd_test$Class, pred_svm_us,curve = TRUE)
x6
plot(x6)


#recall

a6=(pr.curve(fd_test$Class, pred_svm_us,curve = TRUE))
a6
plot(a6)
```

```
Confusion Matrix and Statistics


pred_svm_us     0     1
          0 54871     7
          1  1992    91

               Accuracy : 0.9649
                 95% CI : (0.9634, 0.9664)
    No Information Rate : 0.9983
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0804
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.96497
            Specificity : 0.92857
         Pos Pred Value : 0.99987
         Neg Pred Value : 0.04369
             Prevalence : 0.99828
         Detection Rate : 0.96331
   Detection Prevalence : 0.96343
      Balanced Accuracy : 0.94677

       'Positive' Class : 0


  ROC curve

    Area under curve:
     0.4825758

    Curve for scores from  1  to  2
    ( can be plotted with plot(x) )
```




```
Precision-recall curve

  Area under curve (Integral):
   0.4723244

  Area under curve (Davis & Goadrich):
   0.4723244

  Curve for scores from  1  to  2
  ( can be plotted with plot(x) )
```

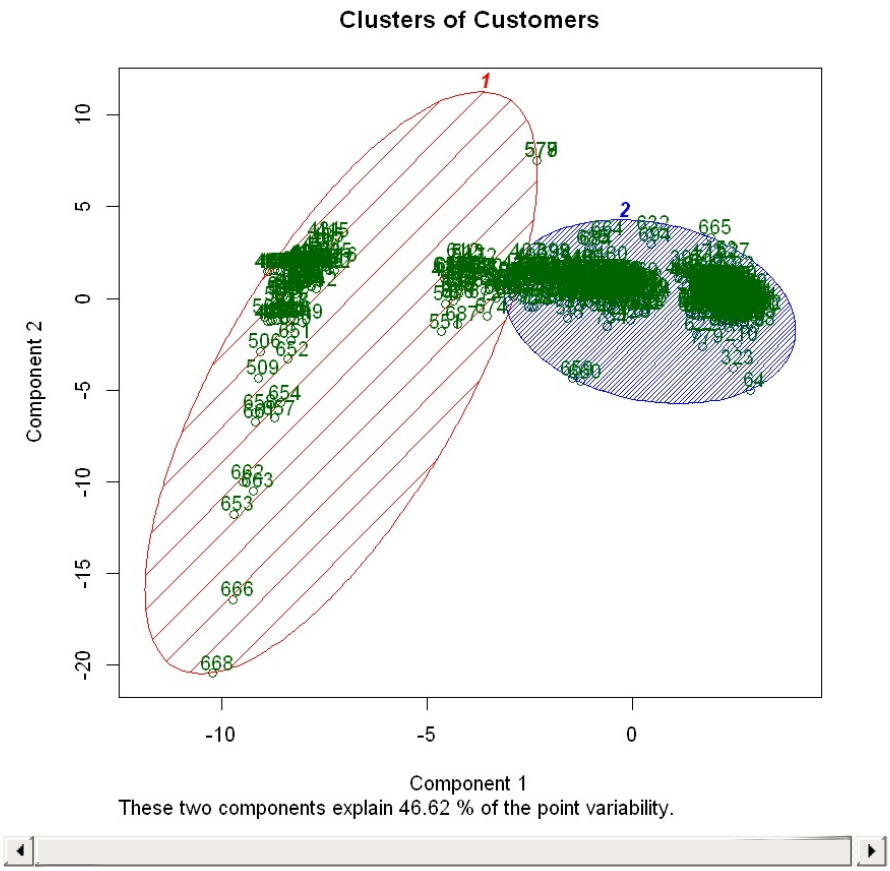

ROC curve
AUC = 0.4825758
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0



In [36]:

```
#Clustering

fdk_us=fd_balanced_under[,c(-29,-30)]

kmeans = kmeans(x = fdk_us, centers = 2)
y_kmeans =(kmeans$cluster)

library(cluster)
clusplot(fd_balanced_under,
         y_kmeans,
         lines = 0,
         shade = TRUE,
         color = TRUE,
         labels = 2,
         plotchar = FALSE,
```

```
        span = TRUE,
        main = paste('Clusters of Customers'))
```



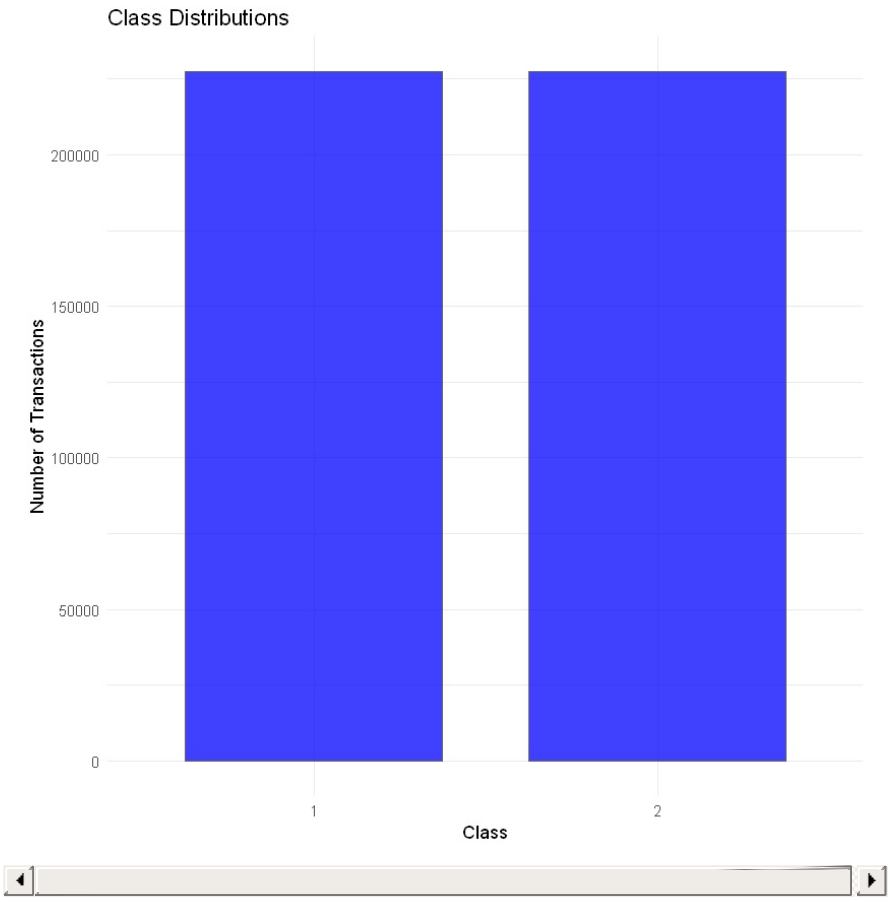
These two components explain 46.62 % of the point variability.

Oversampling

In [39]:

```
##Oversampling

fd_balanced_over = ovun.sample(Class ~ ., data = fd_train, me
thod = "over",N = 2*x11)$data
y112=fd_balanced_over[,c(29,30)]
y112$Class=as.numeric(y112$Class)

#visualising the class
ggplot(y112,aes(x=factor(Class)))+
  geom_bar(stat="count", width=0.75, fill="blue", color = "gr
ey40", alpha = .75)+
  xlab("Class") + ylab("Number of Transactions") +
  ggtitle("Class Distributions") +
  theme_minimal()
table(fd_balanced_over$Class)
```

```
     0      1
227452 227452
```



In [40]:

```
fd_balanced_over=fd_balanced_over[sample(nrow(fd_balanced_over), 1500), ]
```

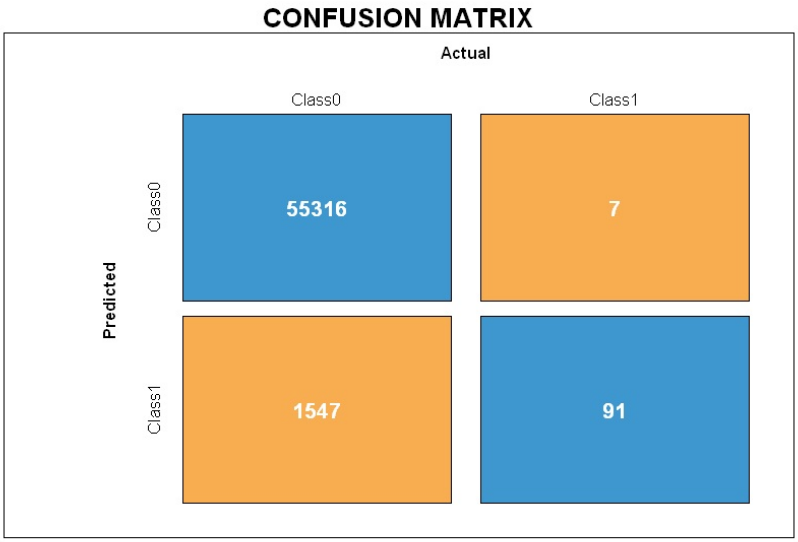
In [41]:

```
#Logistic Regression

classifier_os = glm(formula = Class ~ .,family = binomial,data =fd_balanced_over)
pred_log_os = predict(classifier_os, type = 'response', newdata = fd_test)
pred_log_os_1 = ifelse(pred_log_os > 0.5, 1, 0)

cm7=confusionMatrix(table(pred_log_os_1,fd_test$Class))
cm7
```

```
draw_confusion_matrix(cm7)
x8=roc.curve(fd_test$Class, pred_log_os_1,curve = TRUE)
x8
plot(x8)

a8=(pr.curve(fd_test$Class, pred_log_os_1,curve = TRUE))
a8
plot(a8)
```

```
Confusion Matrix and Statistics


pred_log_os_1     0     1
            0 55316     7
            1  1547    91


               Accuracy : 0.9727182

                 95% CI : (0.9713473, 0.974040
6)
    No Information Rate : 0.9982795

    P-Value [Acc > NIR] : 1


                  Kappa : 0.1019229

 Mcnemar's Test P-Value : <0.0000000000000002


            Sensitivity : 0.97279426

            Specificity : 0.92857143
```

```
         Pos Pred Value : 0.99987347
         Neg Pred Value : 0.05555556
             Prevalence : 0.99827952
         Detection Rate : 0.97112059
   Detection Prevalence : 0.97124348
      Balanced Accuracy : 0.95068284


       'Positive' Class : 0


ROC curve

  Area under curve:
   0.9856464786

  Curve for scores from  0  to  2
  ( can be plotted with plot(x) )
```



DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.973	0.929	1	0.973	0.986

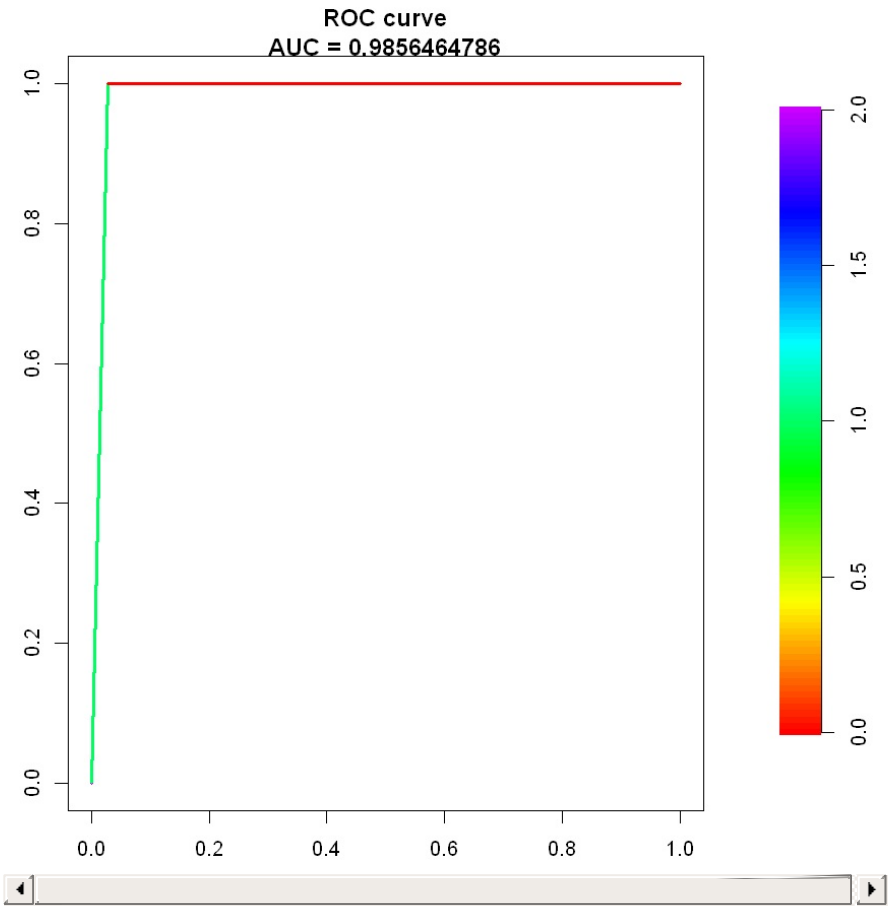
Accuracy	Kappa
0.973	0.102

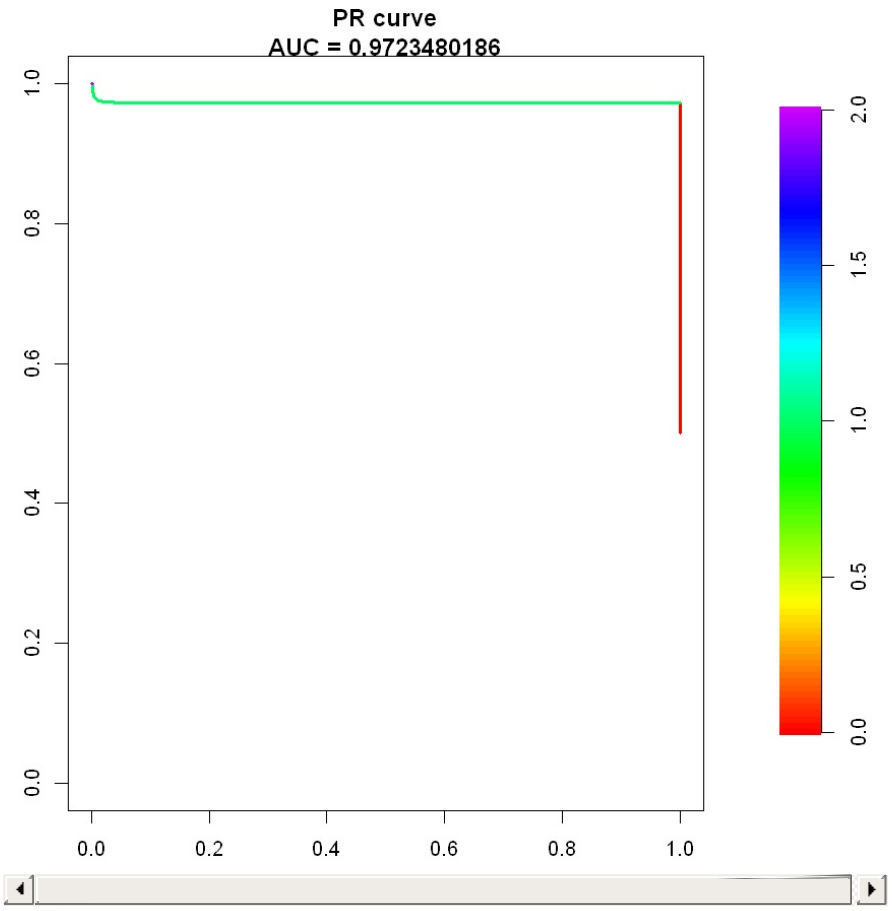
Precision-recall curve

Area under curve (Integral):
0.9723480186

Area under curve (Davis & Goadrich):
0.972348019

Curve for scores from 0 to 2
(can be plotted with plot(x))


ROC curve
AUC = 0.9856464786
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
2.0



In [43]:

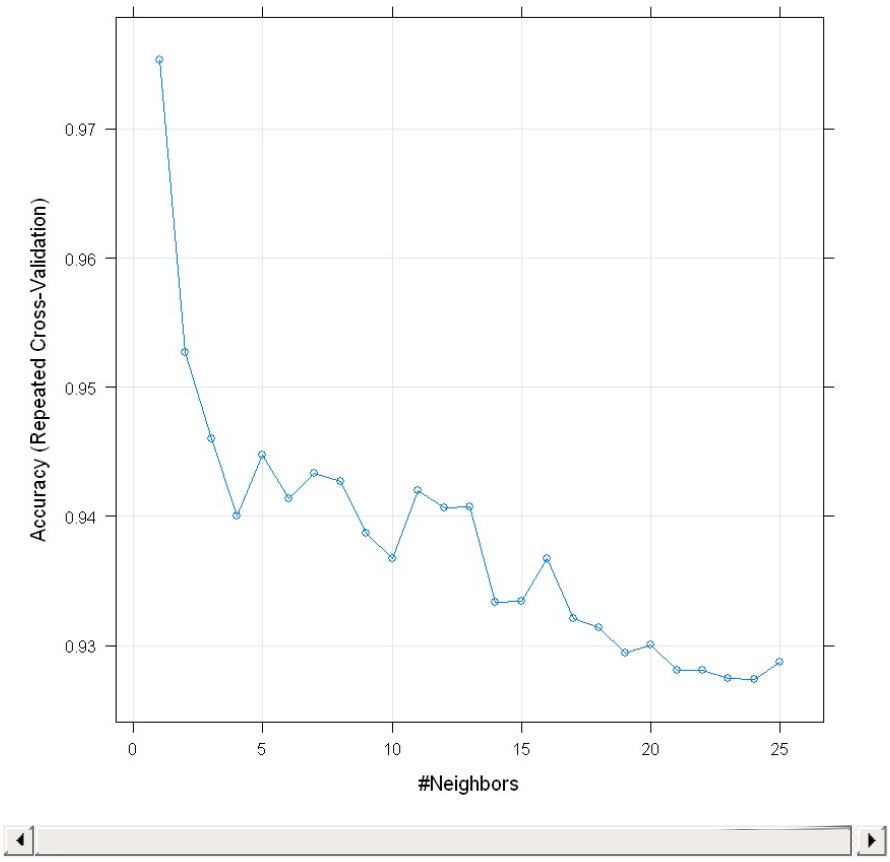
```
#KNN
#hyperparameter tuning

knn_os <- train(Class~., data=fd_balanced_over, method='knn',
  tuneGrid=expand.grid(.k=1:25),metric='Accuracy',
  trControl=trainControl(method='repeatedcv', number=10,repea
ts=1))
knn_os
knn_os_df=as.data.frame(knn_os$results)
knn_os_optimal=max(knn_os_df$k)
plot(knn_os)

#confusion Matrix
pred_knn_os = knn(train = fd_balanced_over[,-29],test = fd_te
st[,-29],
```

```
                  cl = fd_balanced_over[,29],
                  k = knn_os_optimal,
                  prob = TRUE,use.all = F)

cm8=confusionMatrix(table(pred_knn_os,fd_test$Class))
cm8
draw_confusion_matrix(cm8)
x9=roc.curve(fd_test$Class, pred_knn_os,curve = TRUE)
x9
plot(x9)
a9=(pr.curve(fd_test$Class, pred_knn_os,curve = TRUE))
a9
plot(a9)
```

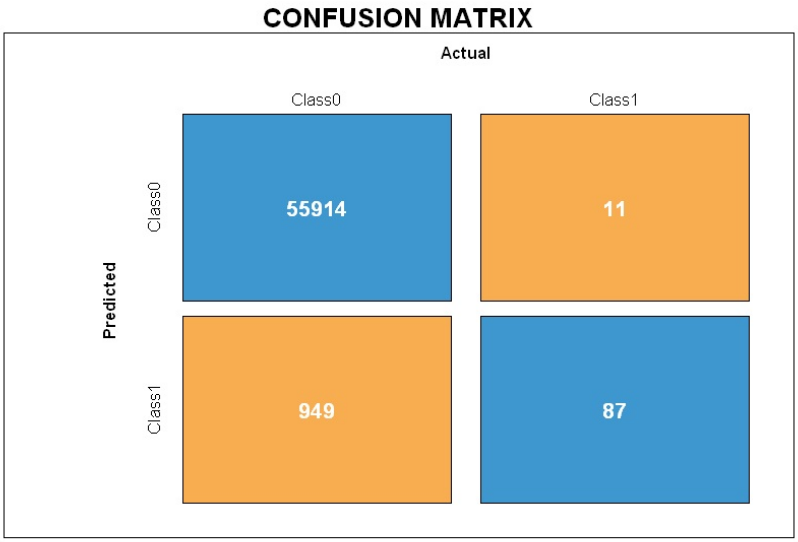
```
k-Nearest Neighbors

1500 samples
  30 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated
 1 times)
Summary of sample sizes: 1349, 1350, 1351, 135
1, 1349, 1350, ...
Resampling results across tuning parameters:

  k   Accuracy      Kappa
   1  0.9753106657  0.9506234162
   2  0.9526701335  0.9053411245
   3  0.9460078226  0.8920234511
   4  0.9400564470  0.8801048299
   5  0.9447187579  0.8894228180
   6  0.9413587567  0.8826897086
   7  0.9433499267  0.8866711335
   8  0.9426744300  0.8853182997
   9  0.9386920900  0.8773452010
```

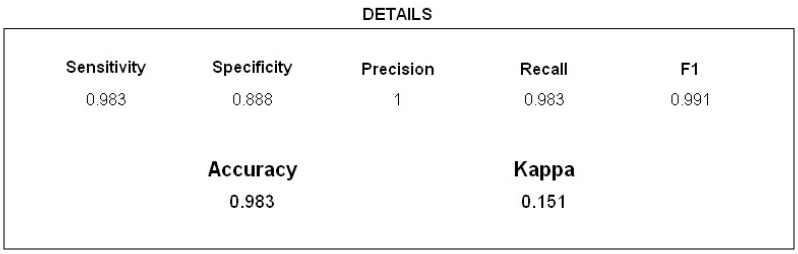
```
  10  0.9366875565  0.8733290623
  11  0.9420299569  0.8840191637
  12  0.9406922678  0.8813457859
  13  0.9407011571  0.8813599694
  14  0.9333675867  0.8666803201
  15  0.9333852468  0.8667040106
  16  0.9367186986  0.8733722760
  17  0.9320474984  0.8640274497
  18  0.9313852468  0.8627058912
  19  0.9293808910  0.8587000195
  20  0.9300519727  0.8600380389
  21  0.9280652177  0.8560633093
  22  0.9280696327  0.8560724328
  23  0.9274074403  0.8547402124
  24  0.9274030253  0.8547295418
  25  0.9287318844  0.8573838280

Accuracy was used to select the optimal model
using the largest value.
The final value used for the model was k = 1.

Confusion Matrix and Statistics


pred_knn_os     0     1
          0 55914    11
          1   949    87


               Accuracy : 0.9831464

                 95% CI : (0.9820555, 0.984187
6)
    No Information Rate : 0.9982795

    P-Value [Acc > NIR] : 1
```

```
                 Kappa : 0.1507695

Mcnemar's Test P-Value : <0.0000000000000002


            Sensitivity : 0.98331076

            Specificity : 0.88775510

         Pos Pred Value : 0.99980331

         Neg Pred Value : 0.08397683

             Prevalence : 0.99827952

         Detection Rate : 0.98161900

   Detection Prevalence : 0.98181212

      Balanced Accuracy : 0.93553293


       'Positive' Class : 0
```




```
ROC curve

  Area under curve:
   0.4917662962

  Curve for scores from  1  to  2
  ( can be plotted with plot(x) )
```



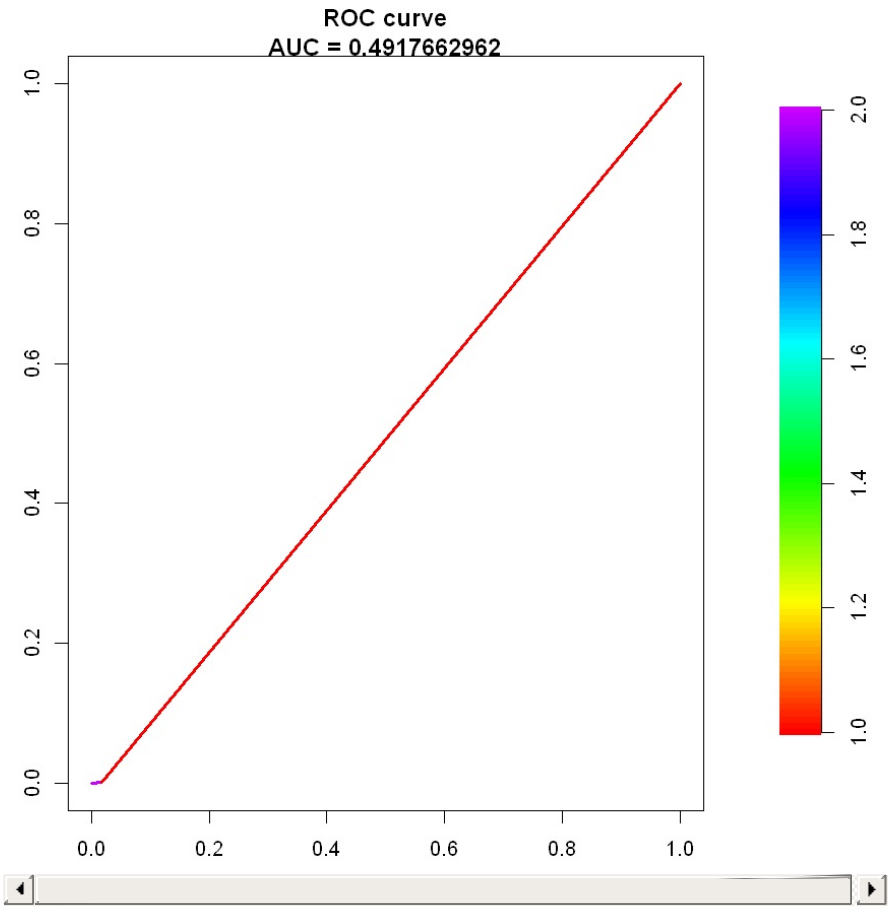


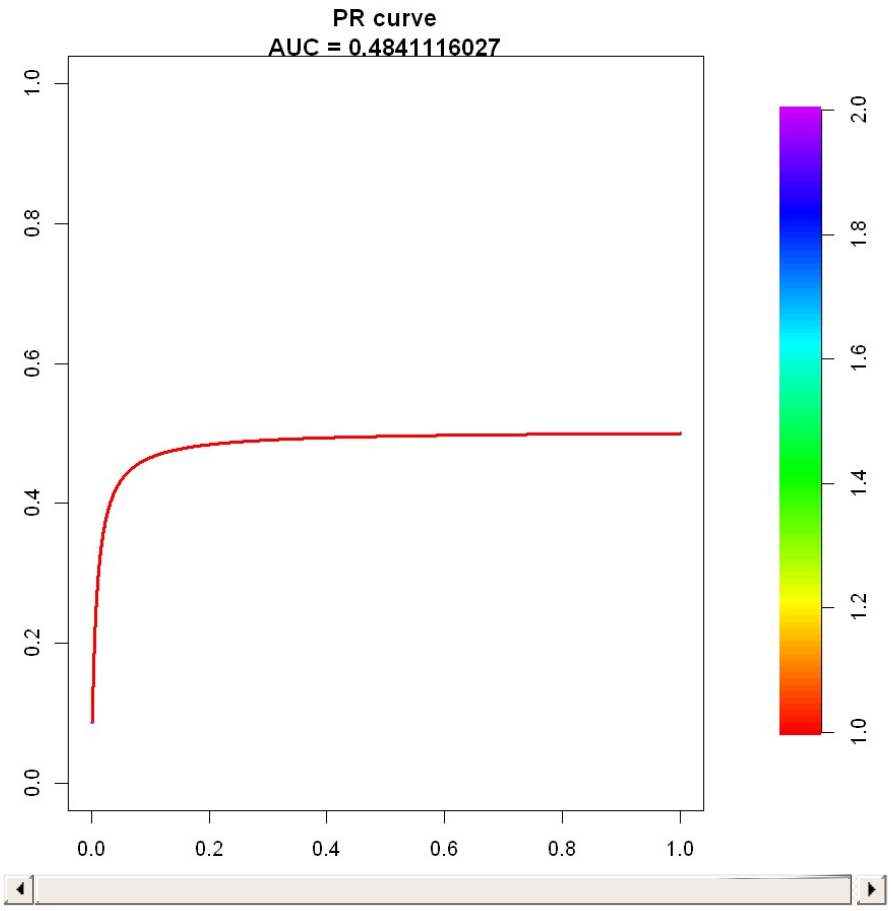
Precision-recall curve

Area under curve (Integral):
0.4841116027

Area under curve (Davis & Goadrich):
0.4841116017

Curve for scores from 1 to 2
(can be plotted with plot(x))


ROC curve
AUC = 0.4917662962
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0



In [44]:

```
#Naive Bayes
#tuning
options(warn=-1)
nb_os = train(x = fd_balanced_over[-29],
              y = fd_balanced_over$Class,method = "nb",
              trControl = trainControl(method='repeatedcv', n
umber=10,repeats=1),
              tuneGrid = expand.grid(usekernel = c(TRUE, FALS
E),fL = 0:3,adjust =                seq(0, 3, by = 1)))
nb_os
```

```
Naive Bayes

1500 samples
  30 predictor
```

```
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated
 1 times)
Summary of sample sizes: 1351, 1350, 1350, 134
9, 1349, 1350, ...
Resampling results across tuning parameters:

  usekernel  fL  adjust  Accuracy      Kappa

  FALSE      0   0       0.9159895106  0.83190
28199
  FALSE      0   1       0.9159895106  0.83190
28199
  FALSE      0   2       0.9159895106  0.83190
28199
  FALSE      0   3       0.9159895106  0.83190
28199
  FALSE      1   0       0.9159895106  0.83190
28199
  FALSE      1   1       0.9159895106  0.83190
28199
  FALSE      1   2       0.9159895106  0.83190
28199
  FALSE      1   3       0.9159895106  0.83190
28199
  FALSE      2   0       0.9159895106  0.83190
28199
  FALSE      2   1       0.9159895106  0.83190
28199
  FALSE      2   2       0.9159895106  0.83190
28199
  FALSE      2   3       0.9159895106  0.83190
28199
  FALSE      3   0       0.9159895106  0.83190
28199
```

```
  FALSE      3   1       0.9159895106  0.83190
28199
  FALSE      3   2       0.9159895106  0.83190
28199
  FALSE      3   3       0.9159895106  0.83190
28199
   TRUE      0   0                NaN
  NaN
   TRUE      0   1       0.9046516438  0.80921
74905
   TRUE      0   2       0.9086516438  0.81720
52329
   TRUE      0   3       0.9079895106  0.81586
34969
   TRUE      1   0                NaN
  NaN
   TRUE      1   1       0.9046516438  0.80921
74905
   TRUE      1   2       0.9086516438  0.81720
52329
   TRUE      1   3       0.9079895106  0.81586
34969
   TRUE      2   0                NaN
  NaN
   TRUE      2   1       0.9046516438  0.80921
74905
   TRUE      2   2       0.9086516438  0.81720
52329
   TRUE      2   3       0.9079895106  0.81586
34969
   TRUE      3   0                NaN
  NaN
   TRUE      3   1       0.9046516438  0.80921
74905
   TRUE      3   2       0.9086516438  0.81720
52329
   TRUE      3   3       0.9079895106  0.81586
```

```
34969

Accuracy was used to select the optimal model
using the largest value.
The final values used for the model were fL =
0, usekernel = FALSE and adjust
 = 0.
```

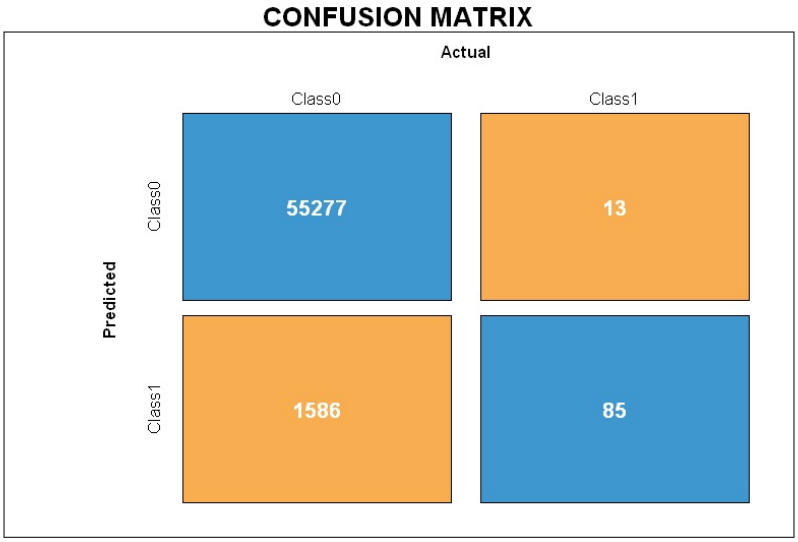
In [45]:

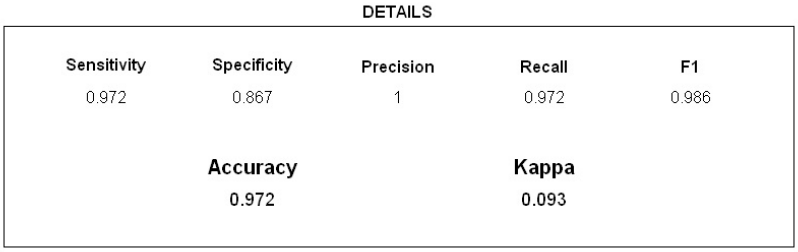
```
#confusion matrix

pred_nb_os = predict(nb_os, newdata = fd_test)
cm9=confusionMatrix(table(pred_nb_os,fd_test$Class))
cm9
draw_confusion_matrix(cm9)
x10=roc.curve(fd_test$Class, pred_nb_os,curve=TRUE)
x10
plot(x10)
a10=(pr.curve(fd_test$Class, pred_nb_os,curve = TRUE))
a10
plot(a10)
```

```
Confusion Matrix and Statistics


pred_nb_os     0     1
         0 55277    13
         1  1586    85


               Accuracy : 0.9719282

                 95% CI : (0.9705386, 0.973269
3)
    No Information Rate : 0.9982795

    P-Value [Acc > NIR] : 1
```

```
                 Kappa : 0.0931519

Mcnemar's Test P-Value : <0.0000000000000002


            Sensitivity : 0.97210840

            Specificity : 0.86734694

         Pos Pred Value : 0.99976488

         Neg Pred Value : 0.05086774

             Prevalence : 0.99827952

         Detection Rate : 0.97043591

   Detection Prevalence : 0.97066414

      Balanced Accuracy : 0.91972767


       'Positive' Class : 0


 ROC curve

    Area under curve:
     0.486192307
```

Curve for scores from 1 to 2
(can be plotted with plot(x))



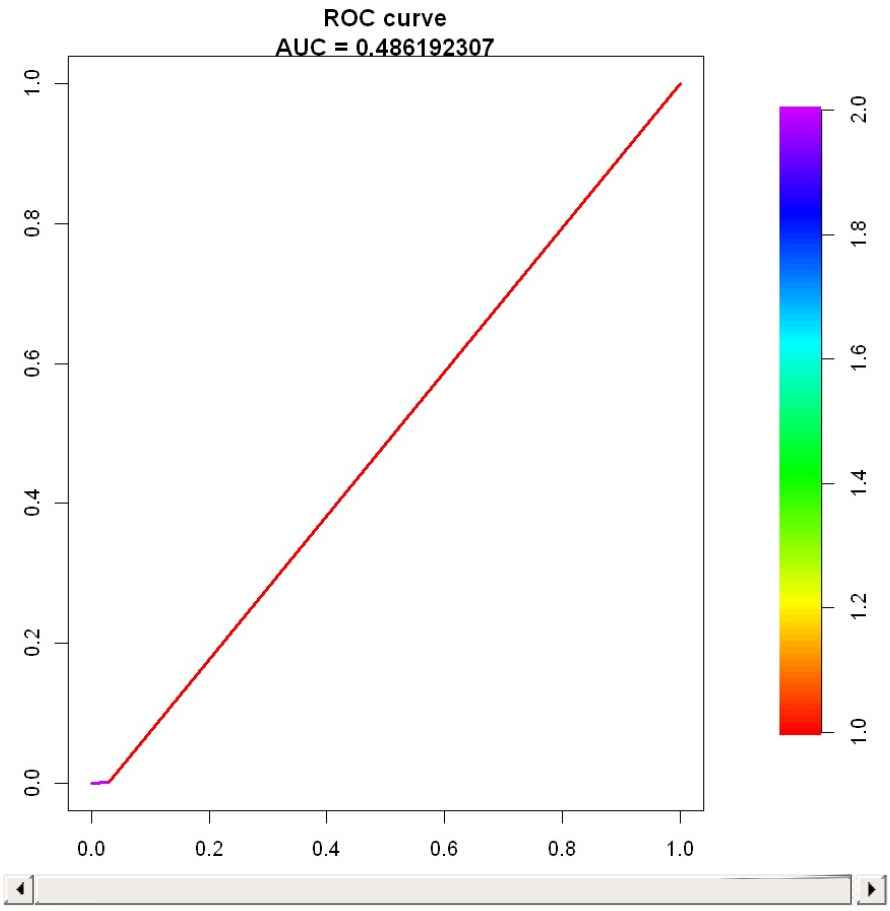


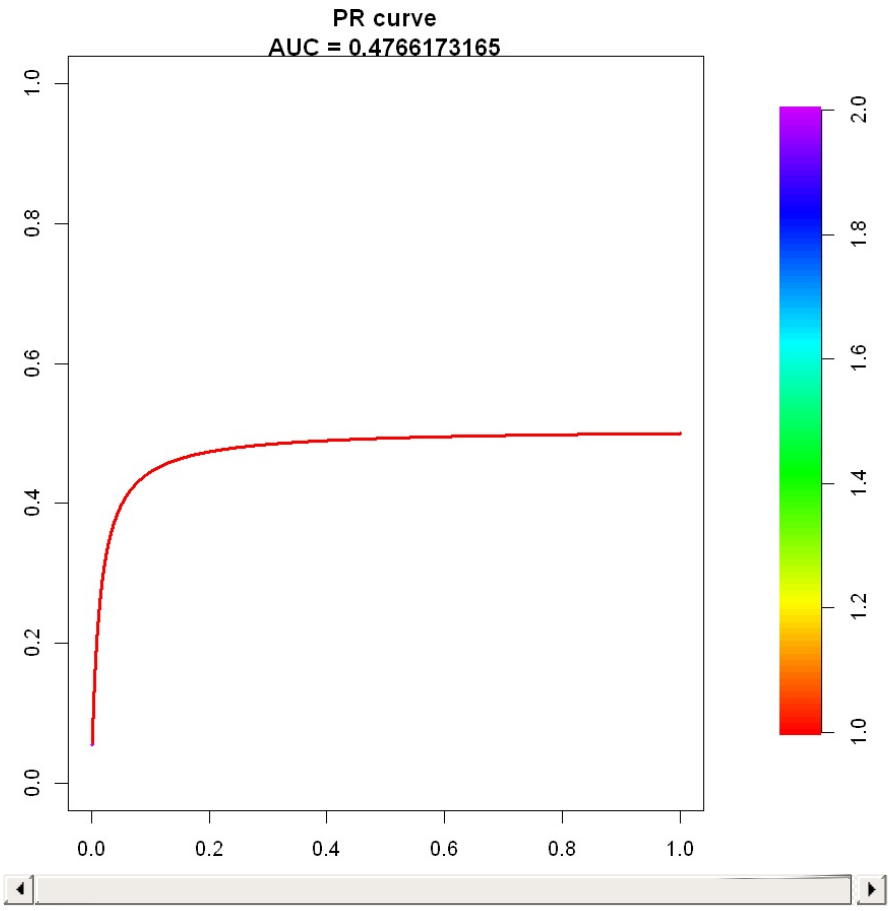
Precision-recall curve

Area under curve (Integral):
 0.4766173165

Area under curve (Davis & Goadrich):
 0.4766173157

Curve for scores from 1 to 2
(can be plotted with plot(x))


ROC curve
AUC = 0.486192307
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0



In [46]:

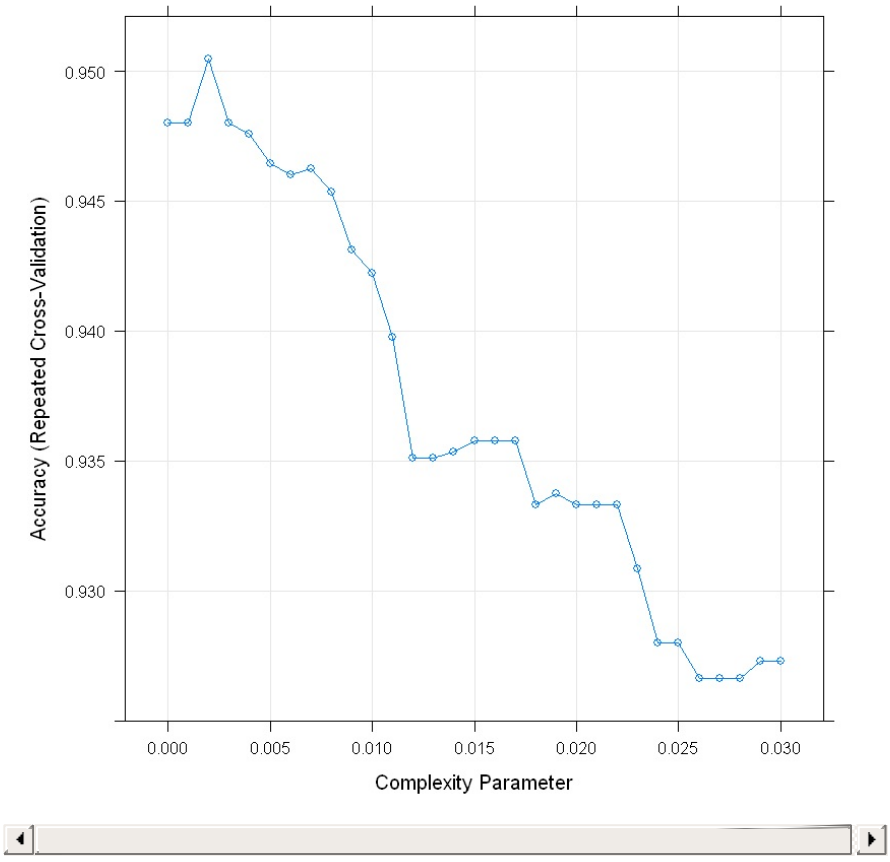
```
#Decision Tree
#tuning
options(warn=-1)
dt_os = train(Class~., data=fd_balanced_over, method='rpart',
              tuneGrid=expand.grid(.cp=seq(0.00,0.03,0.001)),
metric='Accuracy',
              trControl=trainControl(method='repeatedcv', num
ber=10,repeats=3))
dt_os
plot(dt_os)
```

```
CART

1500 samples
  30 predictor
```

```
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated
 3 times)
Summary of sample sizes: 1349, 1350, 1350, 135
1, 1350, 1350, ...
Resampling results across tuning parameters:
```

cp	Accuracy	Kappa
0.000	0.9480121783	0.8960277898
0.001	0.9480106869	0.8960225687
0.002	0.9504522473	0.9008984431
0.003	0.9480106869	0.8960069567
0.004	0.9475677141	0.8951172342
0.005	0.9464536597	0.8928847680
0.006	0.9460092350	0.8919958760
0.007	0.9462314572	0.8924397559
0.008	0.9453410966	0.8906577565
0.009	0.9431099061	0.8861916463
0.010	0.9422210172	0.8844138685
0.011	0.9397765135	0.8795180382
0.012	0.9351038219	0.8701636370
0.013	0.9351038219	0.8701636370
0.014	0.9353245725	0.8706050412
0.015	0.9357660736	0.8714876172
0.016	0.9357660736	0.8714876172
0.017	0.9357660736	0.8714876172
0.018	0.9333126806	0.8665748965
0.019	0.9337541817	0.8674571625
0.020	0.9333112286	0.8665699435
0.021	0.9333112286	0.8665699435
0.022	0.9333112286	0.8665699435
0.023	0.9308667842	0.8616763007
0.024	0.9279807596	0.8558854661
0.025	0.9279807596	0.8558854661
0.026	0.9266459151	0.8532130474

```
  0.027  0.9266459151  0.8532123210
  0.028  0.9266459151  0.8532123210
  0.029  0.9273111101  0.8545409082
  0.030  0.9273111101  0.8545409082

Accuracy was used to select the optimal model
using the largest value.
The final value used for the model was cp = 0.
002.
```



In [47]:

```
#Confusion matrix
options(warn=-1)
fd_balanced_over_dt <- ovun.sample(Class ~ ., data = fd_train
_dt, method = "over",
                                   N = 2*x11_dt, seed = 1)$da
ta
```

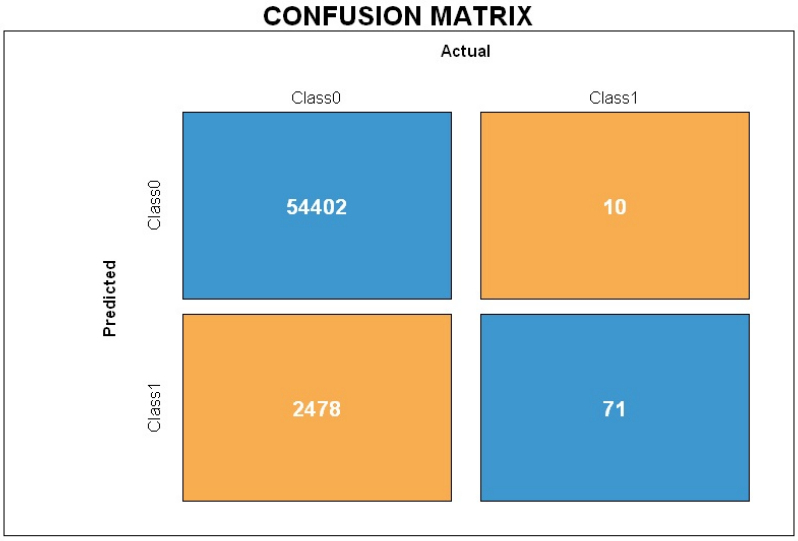
```
fd_balanced_over_dt=fd_balanced_over_dt[sample(nrow(fd_balanc
ed_over_dt), 3000), ]
table(fd_balanced_over_dt$Class)


tree_os = rpart(Class ~ ., data =fd_balanced_over_dt,
                control=rpart.control(cp = 0.0,maxdepth = 8,m
insplit = 100))

prune_os <- prune(tree_os, cp = 0.002)
pred_tree_os <- predict(prune_os, newdata = fd_test_dt)
pred_tree_os_1 = ifelse(pred_tree_os > 0.5, 1, 0)

cm10=confusionMatrix(table(pred_tree_os_1,fd_test_dt$Class))
cm10
draw_confusion_matrix(cm10)
x11_=roc.curve(fd_test_dt$Class, pred_tree_os_1,curve = TRUE)
x11_
plot(x11_)
a11=(pr.curve(fd_test$Class, pred_tree_os_1,curve = TRUE))
a11
plot(a11)
```

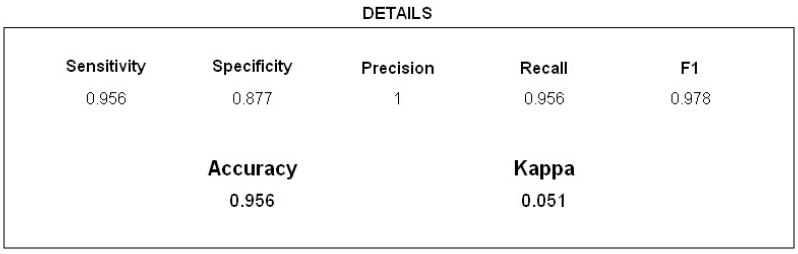
```
   0    1
1495 1505

Confusion Matrix and Statistics


pred_tree_os_1     0     1
             0 54402    10
             1  2478    71


               Accuracy : 0.956321

                 95% CI : (0.9546104, 0.957984
9)
```

```
    No Information Rate : 0.998578
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0513776
 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.95643460
            Specificity : 0.87654321
         Pos Pred Value : 0.99981622
         Neg Pred Value : 0.02785406
             Prevalence : 0.99857797
         Detection Rate : 0.95507452
   Detection Prevalence : 0.95525008
      Balanced Accuracy : 0.91648890

       'Positive' Class : 0
```

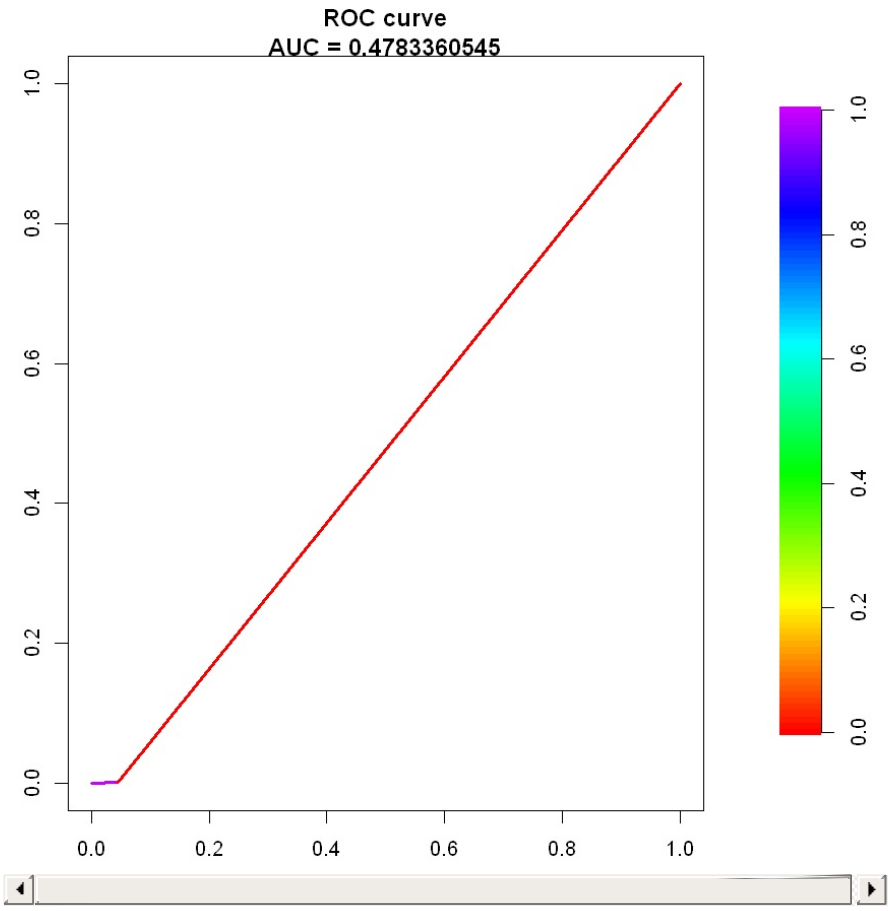
ROC curve

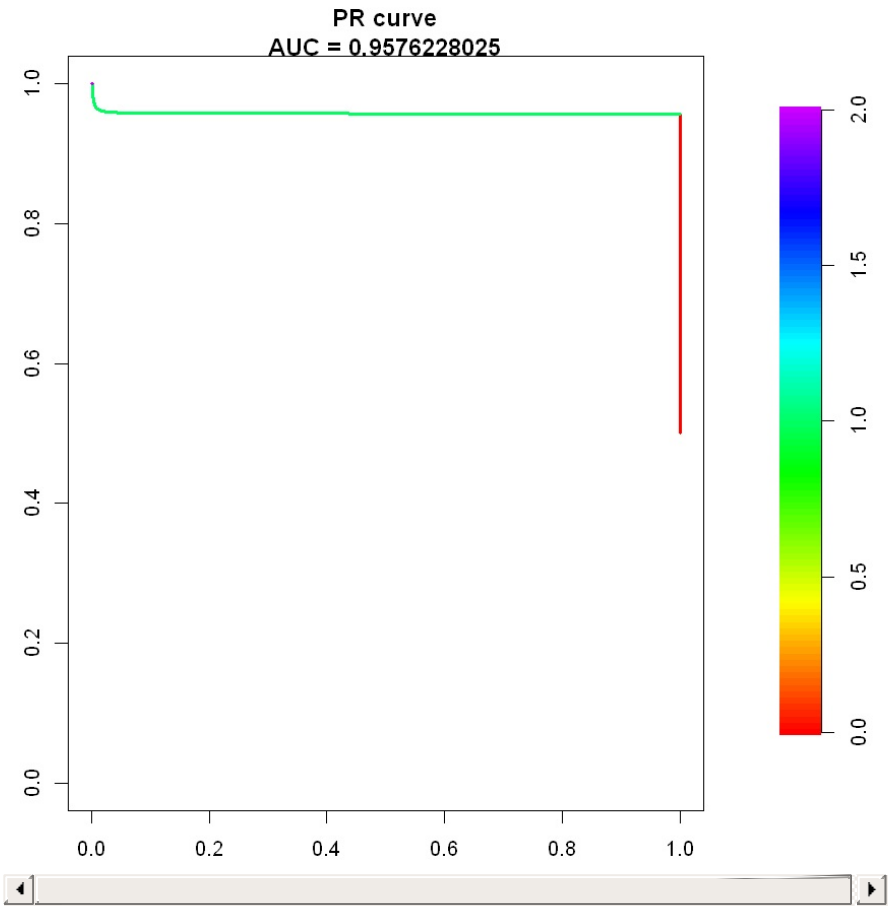
```
Area under curve:
 0.4783360545

Curve for scores from  0  to  1
( can be plotted with plot(x) )
```






```
Precision-recall curve

  Area under curve (Integral):
   0.9576228025

  Area under curve (Davis & Goadrich):
   0.9576228032

  Curve for scores from  0  to  2
  ( can be plotted with plot(x) )
```


ROC curve
AUC = 0.4783360545
0.0
0.2
0.4
0.6
0.8
1.0



In [48]:

```
#Random Forest
#tuning
options(warn=-1)
rf_os = train(Class~., data=fd_balanced_over, method='rf',
               tuneGrid=expand.grid(.mtry=c(1:15)),
              metric='Accuracy',trControl=trainControl(method
='repeatedcv', number=10,repeats=1))
rf_os
```

```
Random Forest

1500 samples
  30 predictor
   2 classes: '0', '1'
```

```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated
 1 times)
Summary of sample sizes: 1349, 1350, 1351, 134
9, 1350, 1351, ...
Resampling results across tuning parameters:

  mtry  Accuracy      Kappa
   1    0.9892930056  0.9785808675
   2    0.9872885313  0.9745724638
   3    0.9866218647  0.9732400984
   4    0.9872885313  0.9745741576
   5    0.9859551387  0.9719067652
   6    0.9853018356  0.9706013056
   7    0.9879641466  0.9759264766
   8    0.9859685023  0.9719346390
   9    0.9859773916  0.9719533732
  10    0.9866396432  0.9732778184
  11    0.9873063099  0.9746109195
  12    0.9853151399  0.9706293926
  13    0.9853151399  0.9706293926
  14    0.9846528883  0.9693056441
  15    0.9859818066  0.9719627259

Accuracy was used to select the optimal model
using the largest value.
The final value used for the model was mtry =
1.
```

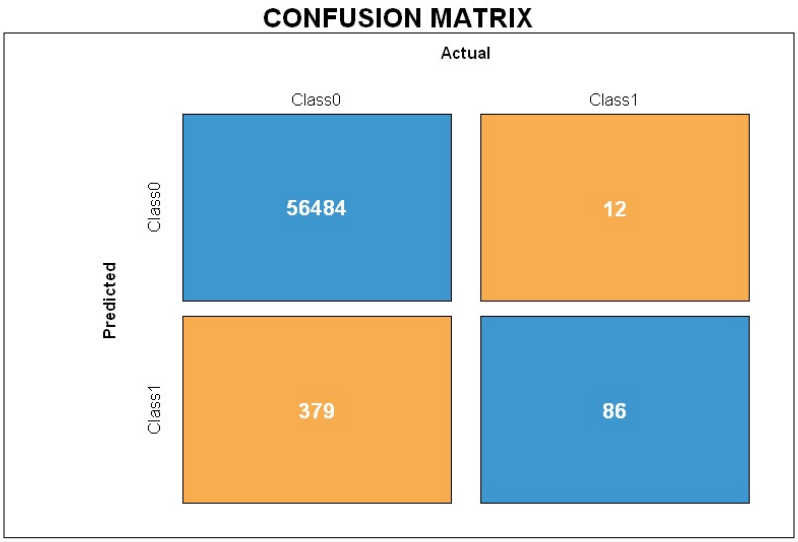
In [49]:

```
#confusion matrix

randomforest_os = randomForest(x = fd_balanced_over[-29],
                               y = fd_balanced_over$Class,
                               mtry = 1)
```

```
pred_rf_os = predict(randomforest_os, newdata = fd_test)

cm11=confusionMatrix(table(pred_rf_os,fd_test$Class))
cm11
draw_confusion_matrix(cm11)
x22_=roc.curve(fd_test$Class, pred_rf_os,curve=TRUE)
x22_
plot(x22_)
a12=(pr.curve(fd_test$Class, pred_nb_os,curve = TRUE))
a12
plot(a12)
```

```
Confusion Matrix and Statistics


pred_rf_os     0     1
         0 56484    12
         1   379    86


               Accuracy : 0.9931357

                 95% CI : (0.9924231, 0.993797
3)
    No Information Rate : 0.9982795

    P-Value [Acc > NIR] : 1


                  Kappa : 0.3035268

 Mcnemar's Test P-Value : <0.0000000000000002


            Sensitivity : 0.9933349
```

```
          Specificity : 0.8775510

       Pos Pred Value : 0.9997876

       Neg Pred Value : 0.1849462

           Prevalence : 0.9982795

       Detection Rate : 0.9916258

 Detection Prevalence : 0.9918365

    Balanced Accuracy : 0.9354429


     'Positive' Class : 0


ROC curve

  Area under curve:
   0.4967784976

  Curve for scores from  1  to  2
  ( can be plotted with plot(x) )
```



DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.993	0.878	1	0.993	0.997

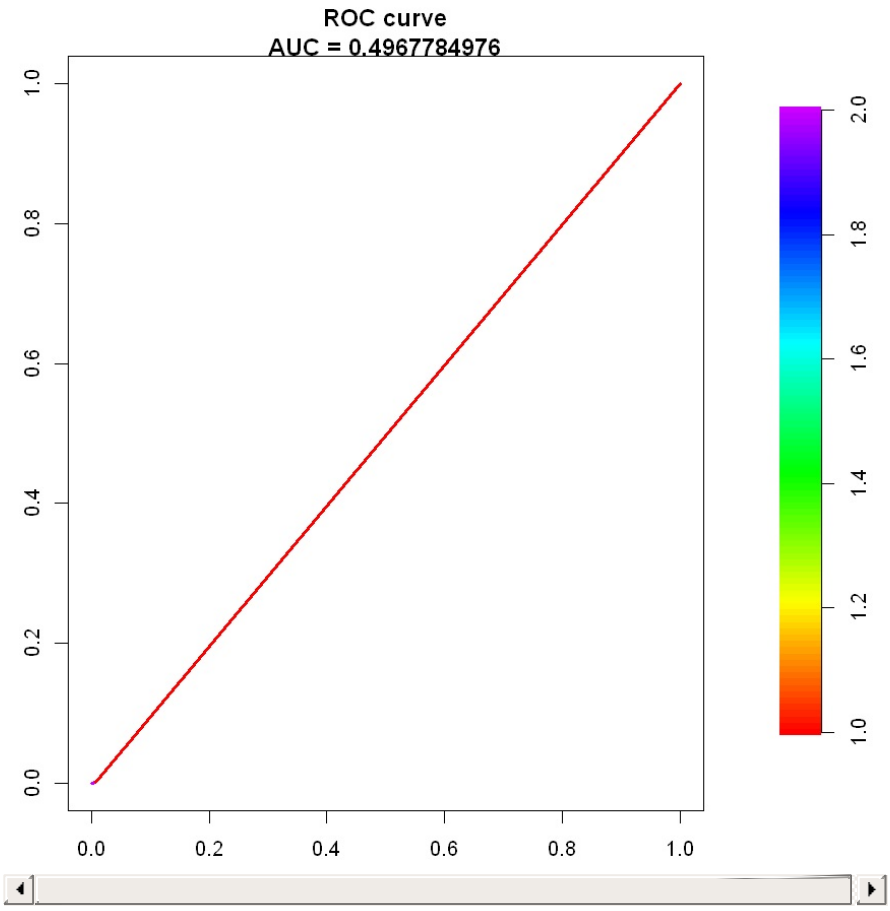
Accuracy	Kappa
0.993	0.304

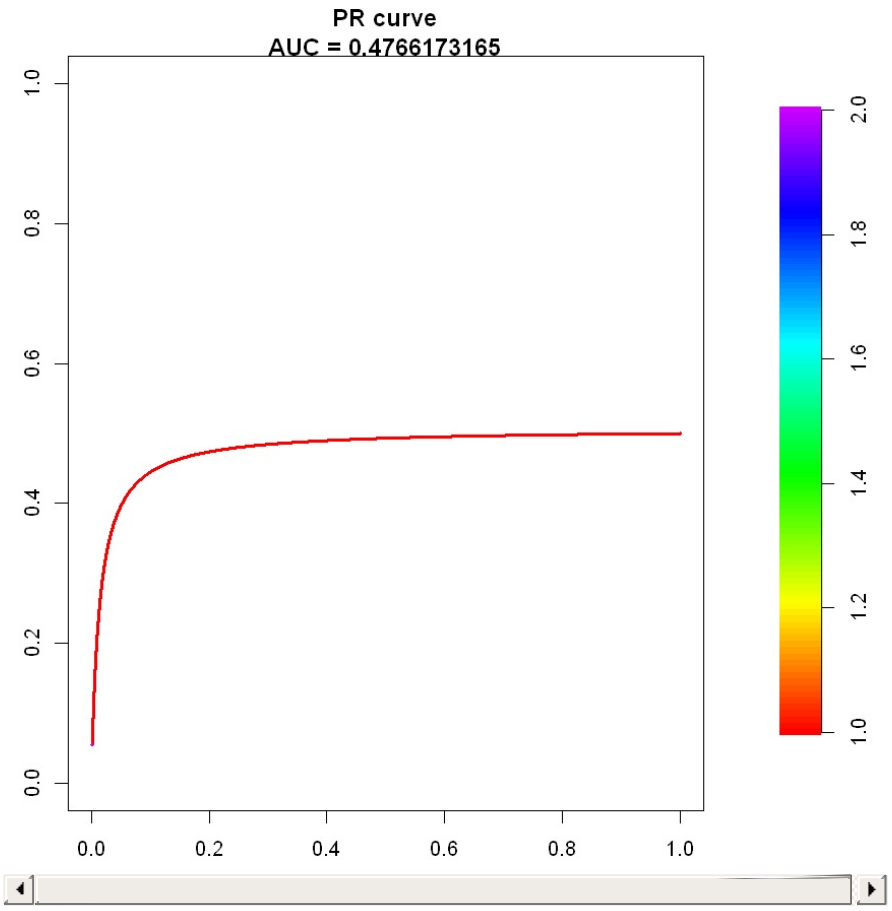
Precision-recall curve

Area under curve (Integral):
0.4766173165

Area under curve (Davis & Goadrich):
0.4766173157

Curve for scores from 1 to 2
(can be plotted with plot(x))


ROC curve
AUC = 0.4967784976
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0



In [50]:

```
#Support Vector Machine
#tuning
options(warn=-1)
svm_tn_os <- train(Class ~., data = fd_balanced_over,
                   method = "svmPoly",
                   trControl=trainControl(method = "repeatedcv",

                                          number = 10, repeats =
 1),
                   preProcess = c("center", "scale"),
                   tuneGrid = expand.grid(.degree = c(2:3),.scal
e = c(0.1,1,10),
                        .C = c(0,0.01, 0.05, 0.1, 0.5,1, 1.5, 2,5
)),
                   tuneLength = 10)
```

```
svm_tn_os
```

```
Support Vector Machines with Polynomial Kernel


1500 samples
  30 predictor
   2 classes: '0', '1'

Pre-processing: centered (30), scaled (30)
Resampling: Cross-Validated (10 fold, repeated
 1 times)
Summary of sample sizes: 1351, 1350, 1350, 134
9, 1351, 1350, ...
Resampling results across tuning parameters:

  degree  scale  C     Accuracy      Kappa

  2        0.1   0.00           NaN
NaN
  2        0.1   0.01  0.9187004459  0.8372850
363
  2        0.1   0.05  0.9393631717  0.8786628
020
  2        0.1   0.10  0.9486923271  0.8973397
184
  2        0.1   0.50  0.9646882084  0.9293588
907
  2        0.1   1.00  0.9640037039  0.9279928
329
  2        0.1   1.50  0.9653592900  0.9307060
907
  2        0.1   2.00  0.9660349053  0.9320569
579
  2        0.1   5.00  0.9660260752  0.9320436
659
  2        1.0   0.00           NaN
```

```
NaN
  2        1.0   0.01  0.9586792598  0.9173399
241
  2        1.0   0.05  0.9653416300  0.9306768
059
  2        1.0   0.10  0.9593460450  0.9186876
303
  2        1.0   0.50  0.9486391099  0.8972745
746
  2        1.0   1.00  0.9492878795  0.8985783
532
  2        1.0   1.50  0.9506346356  0.9012704
087
  2        1.0   2.00  0.9506480584  0.9012973
145
  2        1.0   5.00  0.9493057766  0.8986159
576
  2       10.0   0.00           NaN
NaN
  2       10.0   0.01  0.9419766212  0.8839571
933
  2       10.0   0.05  0.9433277923  0.8866629
897
  2       10.0   0.10  0.9413277331  0.8826665
386
  2       10.0   0.50  0.9413277331  0.8826665
386
  2       10.0   1.00  0.9413277331  0.8826665
386
  2       10.0   1.50  0.9413277331  0.8826665
386
  2       10.0   2.00  0.9413277331  0.8826665
386
  2       10.0   5.00  0.9413277331  0.8826665
386
  3        0.1   0.00           NaN
NaN
```

```
  3         0.1   0.01  0.9306963272  0.8613085
973
  3         0.1   0.05  0.9560124746  0.9119971
358
  3         0.1   0.10  0.9620170674  0.9240143
242
  3         0.1   0.50  0.9673505785  0.9346931
868
  3         0.1   1.00  0.9660172452  0.9320289
024
  3         0.1   1.50  0.9646794969  0.9293550
225
  3         0.1   2.00  0.9680084152  0.9360132
657
  3         0.1   5.00  0.9666839119  0.9333648
965
  3         1.0   0.00           NaN
NaN
  3         1.0   0.01  0.9640081781  0.9280154
329
  3         1.0   0.05  0.9633417485  0.9266877
976
  3         1.0   0.10  0.9620171859  0.9240365
844
  3         1.0   0.50  0.9620171859  0.9240365
844
  3         1.0   1.00  0.9620171859  0.9240365
844
  3         1.0   1.50  0.9620171859  0.9240365
844
  3         1.0   2.00  0.9620171859  0.9240365
844
  3         1.0   5.00  0.9620171859  0.9240365
844
  3        10.0   0.00           NaN
NaN
  3        10.0   0.01  0.9660039409  0.9320115
```

```
469
  3       10.0   0.05  0.9660039409  0.9320115
469
  3       10.0   0.10  0.9660039409  0.9320115
469
  3       10.0   0.50  0.9660039409  0.9320115
469
  3       10.0   1.00  0.9660039409  0.9320115
469
  3       10.0   1.50  0.9660039409  0.9320115
469
  3       10.0   2.00  0.9660039409  0.9320115
469
  3       10.0   5.00  0.9660039409  0.9320115
469

Accuracy was used to select the optimal model
using the largest value.
The final values used for the model were degre
e = 3, scale = 0.1 and C = 2.
```

In [51]:

```
#Tuning and Confusion Matrix

tune_out_os = tune.svm(x = fd_balanced_over[,-29], y = fd_bal
anced_over[,29],
                       type = "C-classification", kernel = "p
olynomial", degree = 3,
                       cost = 2, gamma = c(0.1,1,10), coef0 =
 c(0.1,1,10))
cost1=tune_out$best.parameters$cost
cost1
gamma1=tune_out$best.parameters$gamma
gamma1
coef0_1=tune_out$best.parameters$coef0
coef0_1
```

```
svm_os = svm(Class~ ., data = fd_balanced_over, type = "C-classification",
             kernel = "polynomial", degree = 3,scale = 0.1,
             cost = cost1,
             gamma = gamma1,
             coef0 = coef0_1)

pred_svm_os = predict(svm_os, newdata = fd_test)
cm12=confusionMatrix(table(pred_svm_os,fd_test$Class))
cm12
draw_confusion_matrix(cm12)
x13=roc.curve(fd_test$Class, pred_svm_os,curve =TRUE)
x13
plot(x13)
a13=(pr.curve(fd_test$Class, pred_nb_os,curve = TRUE))
a13
plot(a13)
```

0.5

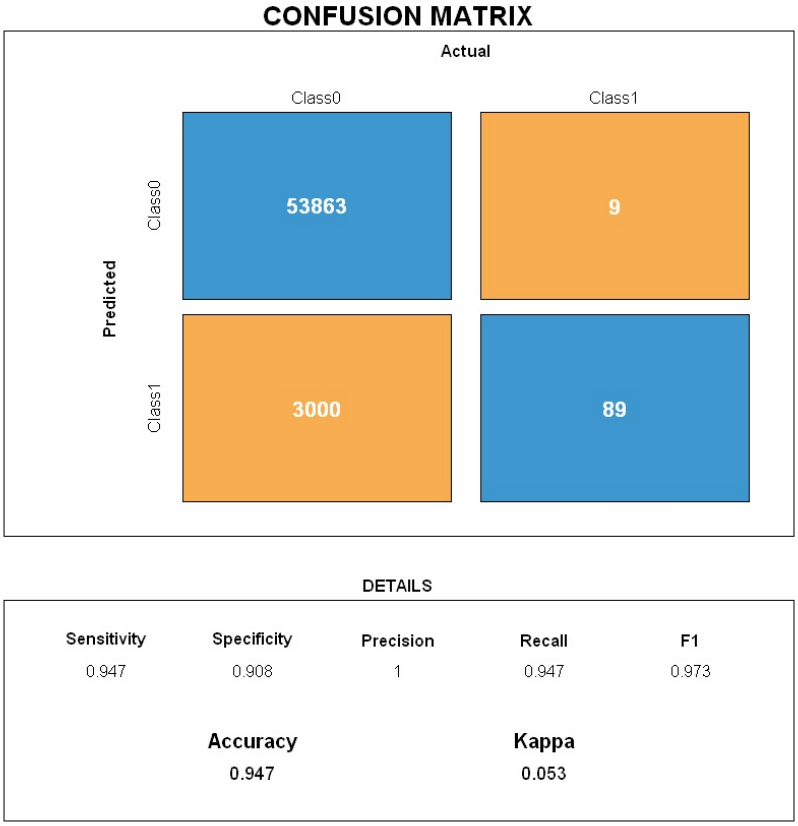
0.1

10

```
Confusion Matrix and Statistics


pred_svm_os     0     1
          0 53863     9
          1  3000    89


               Accuracy : 0.9471744

                 95% CI : (0.9453057, 0.948997
3)
    No Information Rate : 0.9982795
```

```
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0526925

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.94724162

            Specificity : 0.90816327

         Pos Pred Value : 0.99983294

         Neg Pred Value : 0.02881191

             Prevalence : 0.99827952

         Detection Rate : 0.94561191

   Detection Prevalence : 0.94576991

      Balanced Accuracy : 0.92770244

       'Positive' Class : 0


 ROC curve

    Area under curve:
     0.4737451941
```

Curve for scores from 1 to 2
(can be plotted with plot(x))

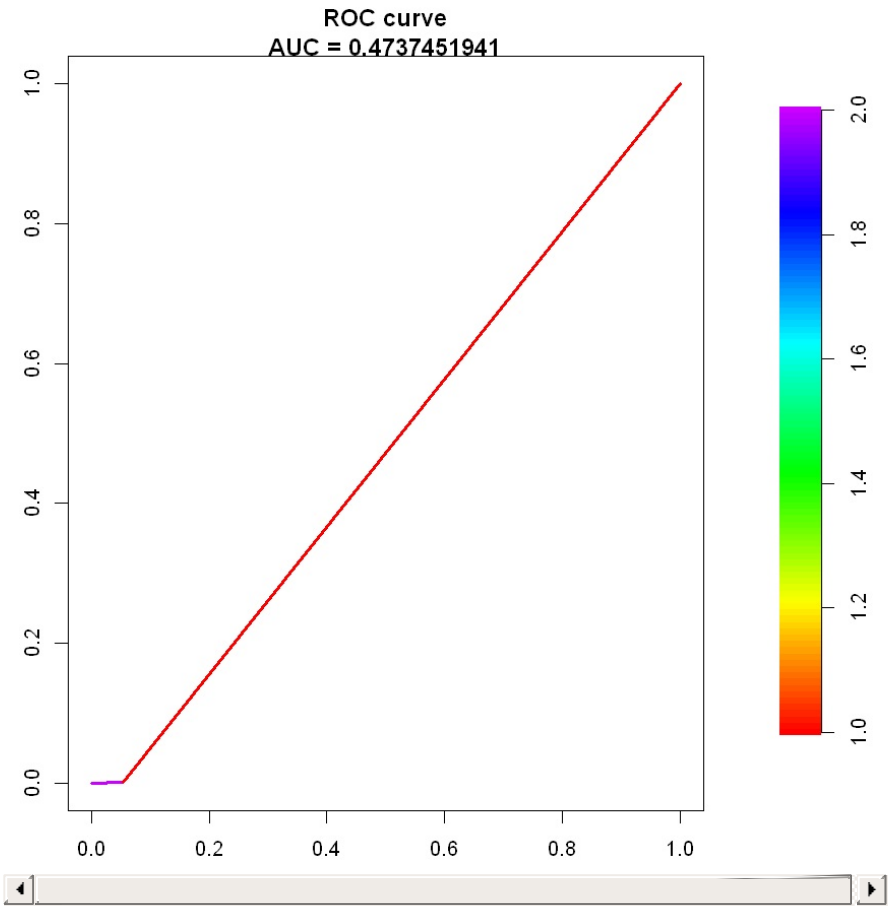


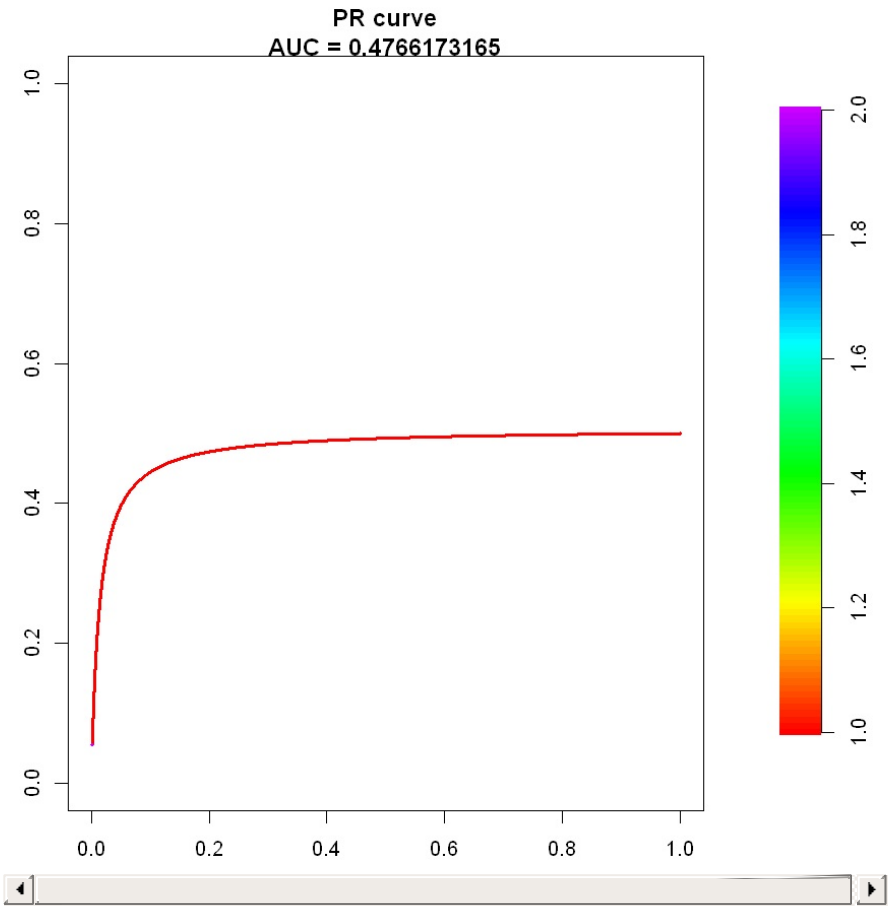
Precision-recall curve

Area under curve (Integral):
 0.4766173165

Area under curve (Davis & Goadrich):
 0.4766173157

Curve for scores from 1 to 2
(can be plotted with plot(x))

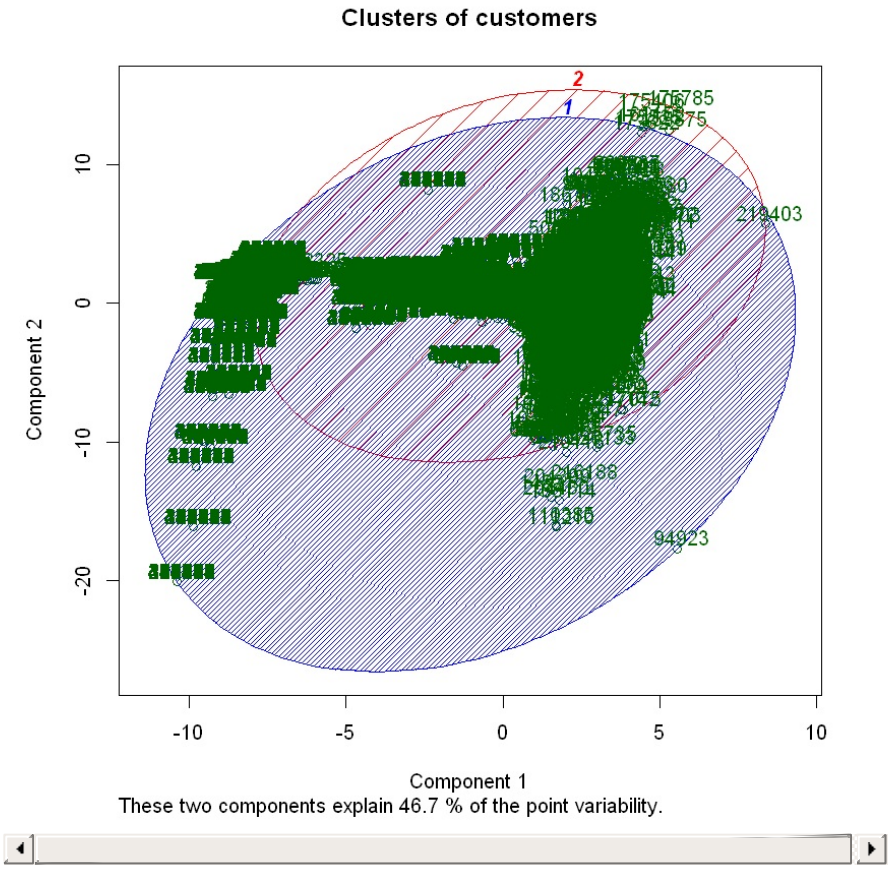

ROC curve
AUC = 0.4737451941
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0



In [53]:

```
#clustering

fd_balanced_over_k = ovun.sample(Class ~ ., data = fd_train,
method = "over",N = 2*x11)$data
fdk_1=fd_balanced_over_k[,c(-29,-30)]

kmeans = kmeans(x = fdk_1, centers = 2)
y_kmeans =(kmeans$cluster)
clusplot(fd_balanced_over_k,
         y_kmeans,
         lines = 0,
         shade = TRUE,
         color = TRUE,
         labels = 2,
         plotchar = FALSE,
```

```
        span = TRUE,
        main = paste('Clusters of customers'))
```

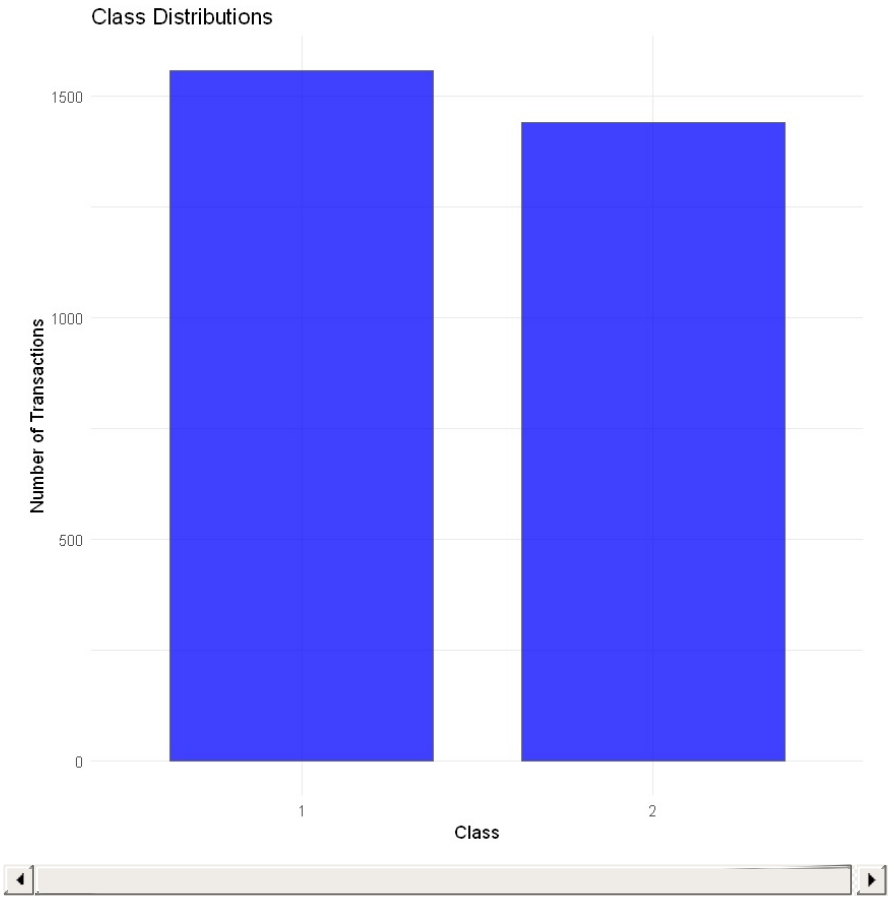


In [55]:

```
#Combined Sampling

fd_balanced_both = ovun.sample(Class ~ ., data =fd_train, met
hod = "both", p=0.5,N=3000, seed = 1)$data
y113=fd_balanced_both[,c(29,30)]
y113$Class=as.numeric(y113$Class)

#visualising the class
ggplot(y113,aes(x=factor(Class)))+
  geom_bar(stat="count", width=0.75, fill="blue", color = "gr
ey40", alpha = .75)+
  xlab("Class") + ylab("Number of Transactions") +
  ggtitle("Class Distributions") +
```

```
  theme_minimal()

table(fd_balanced_both$Class)
```

```
   0    1
1558 1442
```



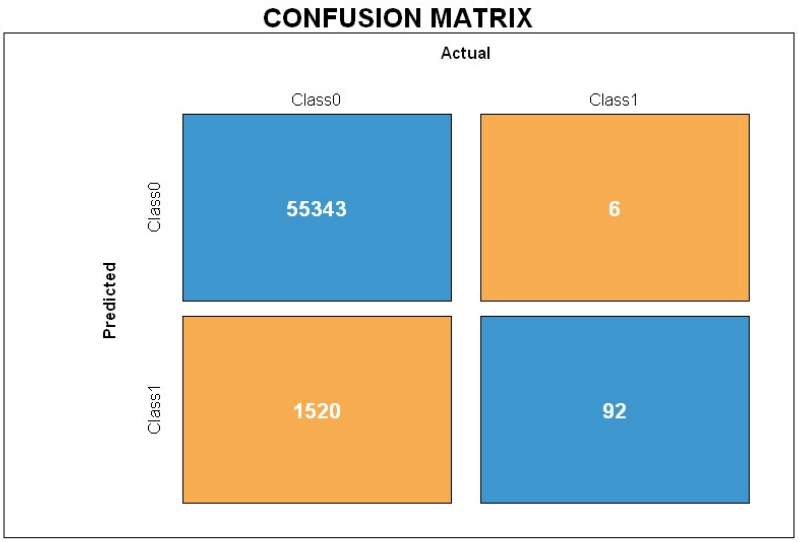
In [57]:

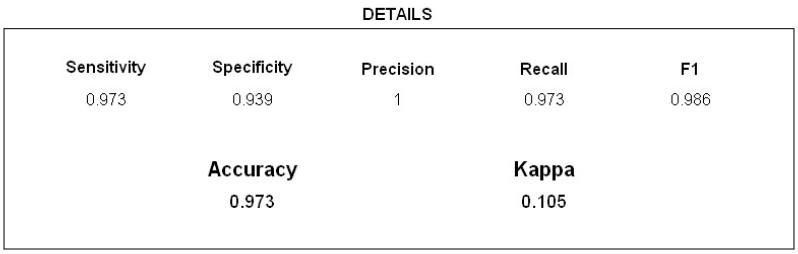
```
#Logistic Regression

classifier_cm = glm(formula = Class ~ .,family = binomial,dat
a =fd_balanced_both)
pred_log_cm = predict(classifier_cm, type = 'response', newda
ta = fd_test)
pred_log_cm_1 = ifelse(pred_log_cm > 0.5, 1, 0)

cm13=confusionMatrix(table(pred_log_cm_1,fd_test$Class))
```

```
cm13
draw_confusion_matrix(cm13)
a15=roc.curve(fd_test$Class, pred_log_cm_1,curve=TRUE)
a15
plot(a15)
x13=(pr.curve(fd_test$Class, pred_log_cm_1,curve = TRUE))
x13
plot(x13)
```

```
Confusion Matrix and Statistics


pred_log_cm_1     0     1
            0 55343     6
            1  1520    92


               Accuracy : 0.9732097

                 95% CI : (0.9718506, 0.974520
4)
    No Information Rate : 0.9982795

    P-Value [Acc > NIR] : 1


                  Kappa : 0.1046982

 Mcnemar's Test P-Value : <0.0000000000000002


            Sensitivity : 0.97326909

            Specificity : 0.93877551
```

```
         Pos Pred Value : 0.99989160
         Neg Pred Value : 0.05707196
             Prevalence : 0.99827952
         Detection Rate : 0.97159460
   Detection Prevalence : 0.97169994
      Balanced Accuracy : 0.95602230


       'Positive' Class : 0


ROC curve

  Area under curve:
   0.9858743123

  Curve for scores from  0  to  2
  ( can be plotted with plot(x) )
```



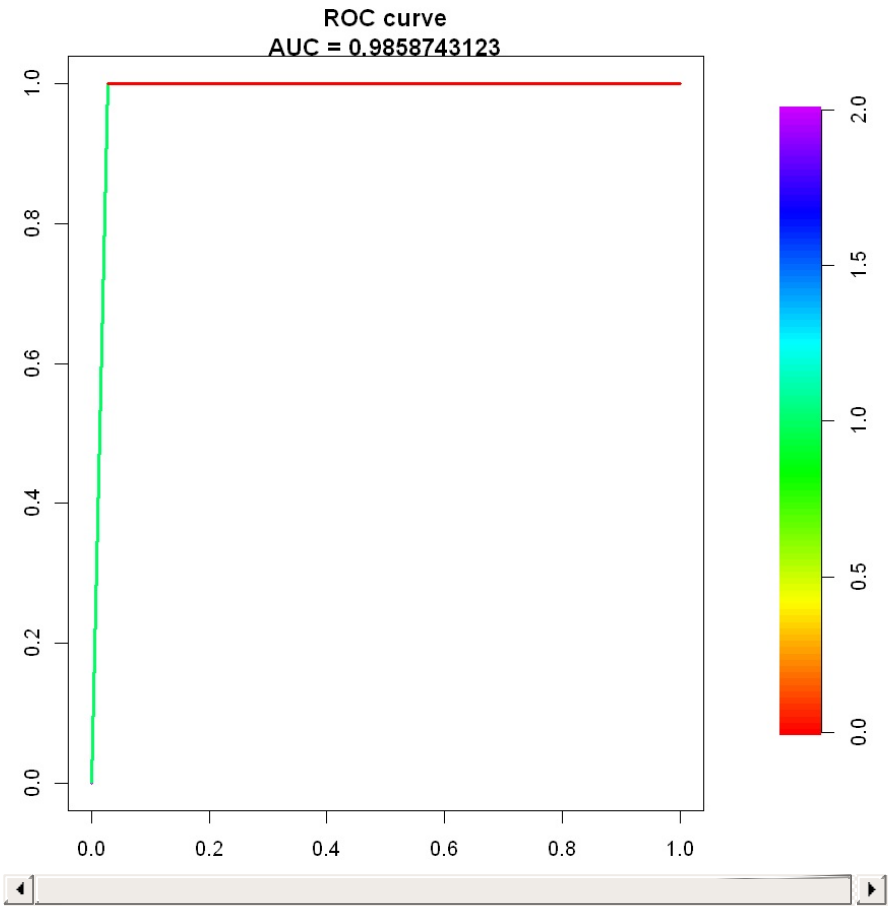


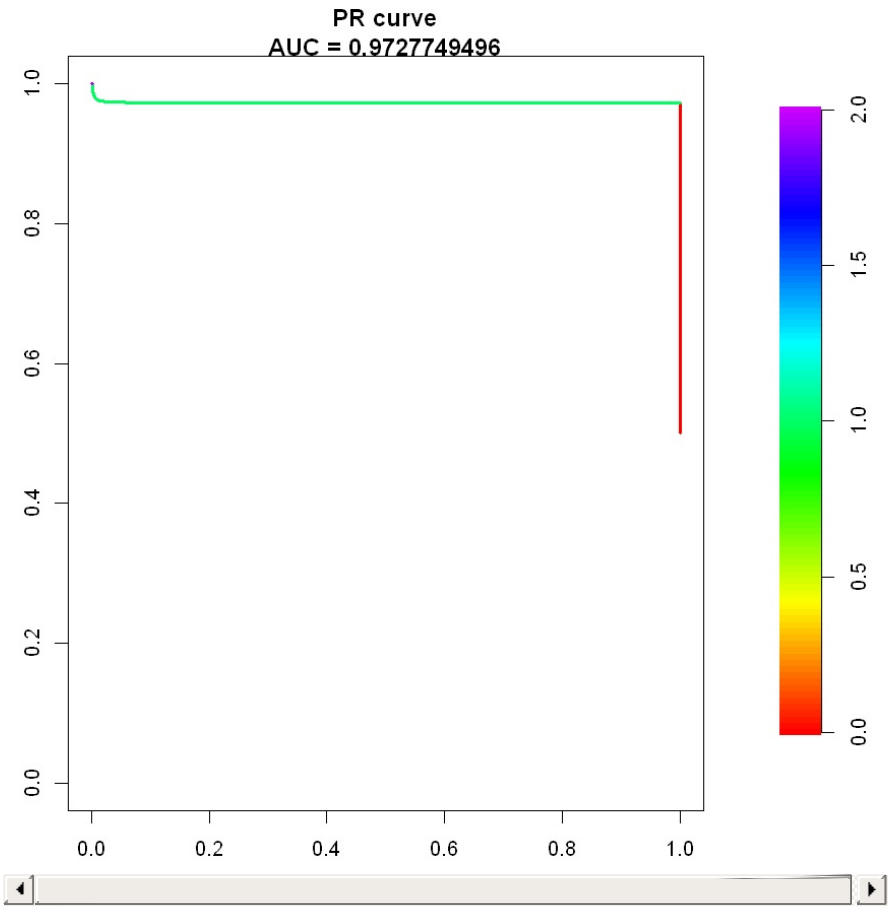
Precision-recall curve

Area under curve (Integral):
0.9727749496

Area under curve (Davis & Goadrich):
0.9727749501

Curve for scores from 0 to 2
(can be plotted with plot(x))


ROC curve
AUC = 0.9858743123
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
2.0



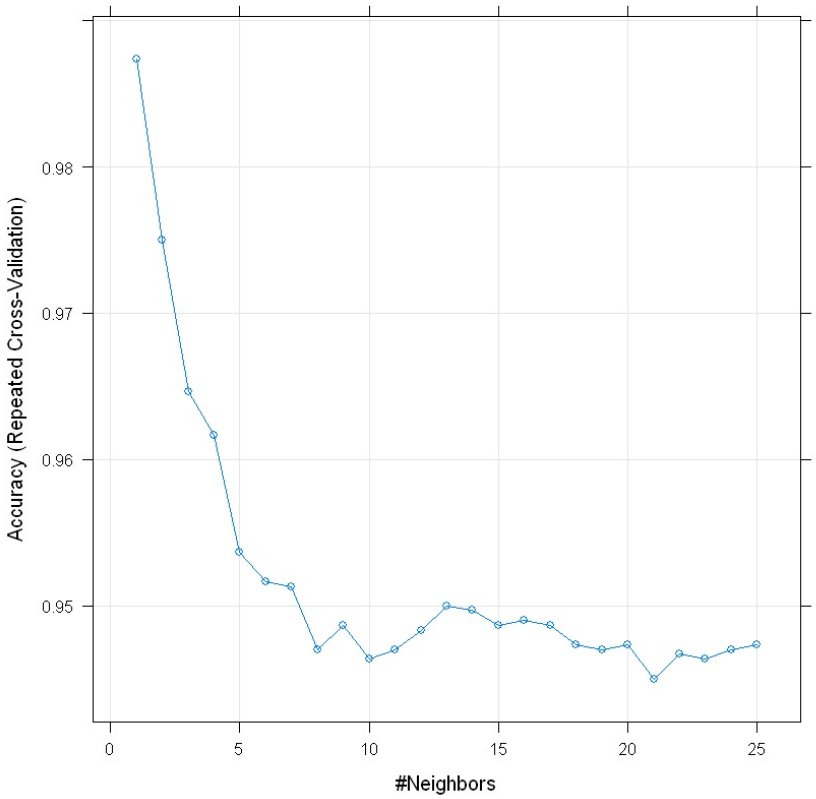
In [58]:

```
#KNN
#tuning
knn_cm <- train(Class~., data=fd_balanced_both, method='knn',
                tuneGrid=expand.grid(.k=1:25),metric='Accurac
y',
                trControl=trainControl(method='repeatedcv', n
umber=10,repeats=1))
knn_cm
knn_cm_df=as.data.frame(knn_cm$results)
knn_cm_optimal=max(knn_cm_df$k)
plot(knn_cm)
```

```
k-Nearest Neighbors

3000 samples
```

```
  30 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated
 1 times)
Summary of sample sizes: 2700, 2700, 2701, 270
0, 2699, 2700, ...
Resampling results across tuning parameters:

  k   Accuracy      Kappa
   1  0.9873343519  0.9746570087
   2  0.9750076334  0.9500120219
   3  0.9646742630  0.9293364566
   4  0.9616642444  0.9232917792
   5  0.9536630926  0.9072881021
   6  0.9516686445  0.9032592482
   7  0.9513342111  0.9025732661
   8  0.9470041926  0.8938936717
   9  0.9486697667  0.8971600381
  10  0.9463408482  0.8924334584
  11  0.9470086371  0.8937468116
  12  0.9483397704  0.8964003377
  13  0.9500053297  0.8997518060
  14  0.9496753334  0.8990818774
  15  0.9486708741  0.8970584558
  16  0.9490086594  0.8977183894
  17  0.9486753334  0.8970489057
  18  0.9473430927  0.8943796442
  19  0.9470142039  0.8937016948
  20  0.9473453149  0.8943366305
  21  0.9450130668  0.8896363538
  22  0.9466841854  0.8929726125
  23  0.9463530743  0.8922949763
  24  0.9470197336  0.8936226961
  25  0.9473541817  0.8942938168
```

```
Accuracy was used to select the optimal model
using the largest value.
The final value used for the model was k = 1.
```



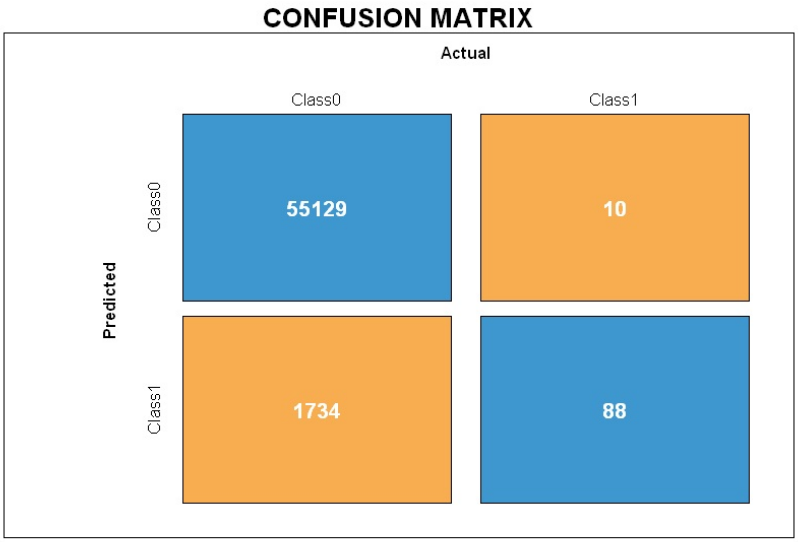
In [59]:

```
#Confusion Matrix

pred_knn_cm = knn(train = fd_balanced_both[,-29],test = fd_te
st[,-29],
                  cl = fd_balanced_both[,29],
                  k = knn_cm_optimal,
                  prob = TRUE)

cm14=confusionMatrix(table(pred_knn_cm,fd_test$Class))
cm14
draw_confusion_matrix(cm14)
x16=roc.curve(fd_test$Class, pred_knn_cm,curve = TRUE)
```

```
x16
plot(x16)
a16=(pr.curve(fd_test$Class, pred_nb_os,curve = TRUE))
a16
plot(a16)
```

```
Confusion Matrix and Statistics


pred_knn_cm     0     1
          0 55129    10
          1  1734    88


               Accuracy : 0.9693826

                 95% CI : (0.9679349, 0.970782
2)
    No Information Rate : 0.9982795

    P-Value [Acc > NIR] : 1


                  Kappa : 0.088691

 Mcnemar's Test P-Value : <0.0000000000000002


            Sensitivity : 0.96950565

            Specificity : 0.89795918

         Pos Pred Value : 0.99981864

         Neg Pred Value : 0.04829857
```

```
            Prevalence : 0.99827952

        Detection Rate : 0.96783764

  Detection Prevalence : 0.96801320

     Balanced Accuracy : 0.93373242


      'Positive' Class : 0


ROC curve

  Area under curve:
   0.4848668387

  Curve for scores from  1  to  2
  ( can be plotted with plot(x) )
```



DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.97	0.898	1	0.97	0.984

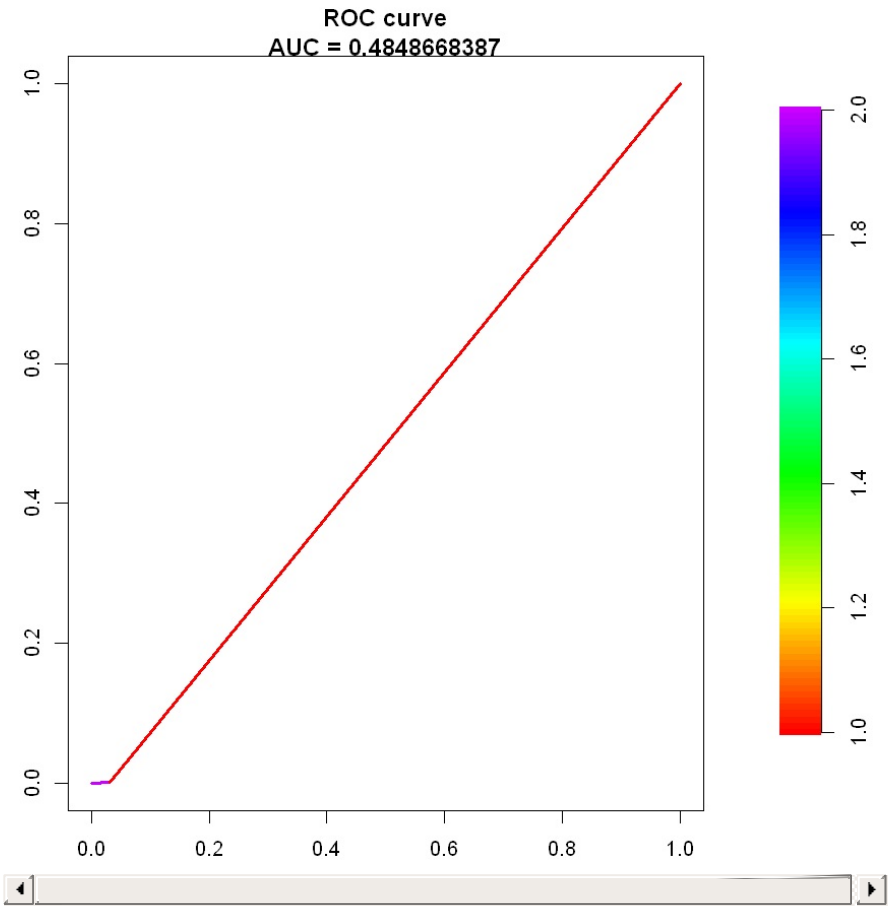
Accuracy	Kappa
0.969	0.089

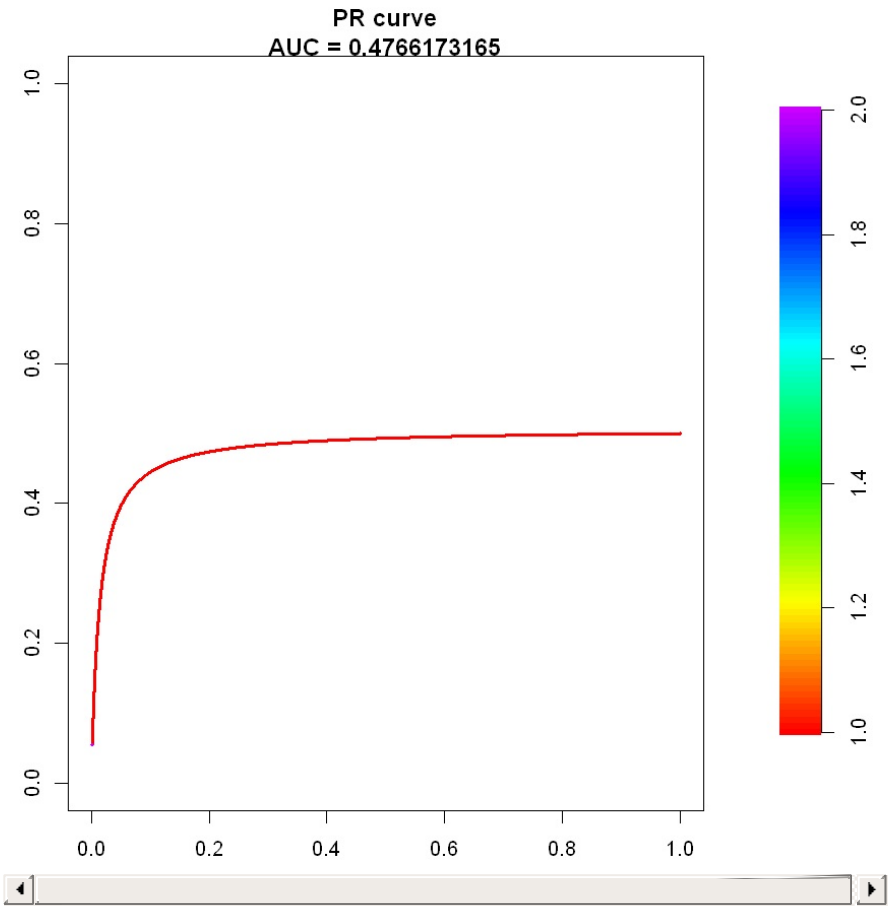
Precision-recall curve

Area under curve (Integral):
0.4766173165

Area under curve (Davis & Goadrich):
0.4766173157

Curve for scores from 1 to 2
(can be plotted with plot(x))


ROC curve
AUC = 0.4848668387
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0



In [60]:

```
#Naive Bayes
#tuning
options(warn=-1)
nb_cm = train(x = fd_balanced_both[-29],
              y = fd_balanced_both$Class,method = "nb",
              trControl = trainControl(method='repeatedcv', n
umber=10,repeats=1),
              tuneGrid = expand.grid(usekernel = c(TRUE, FALS
E),fL = 0:3,adjust =                  seq(0, 3, by = 1)))
nb_cm
```

```
Naive Bayes

3000 samples
  30 predictor
```

```
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated
 1 times)
Summary of sample sizes: 2700, 2700, 2700, 270
0, 2700, 2700, ...
Resampling results across tuning parameters:

  usekernel  fL  adjust  Accuracy      Kappa

  FALSE      0   0       0.9190153557  0.83719
28804
  FALSE      0   1       0.9190153557  0.83719
28804
  FALSE      0   2       0.9190153557  0.83719
28804
  FALSE      0   3       0.9190153557  0.83719
28804
  FALSE      1   0       0.9190153557  0.83719
28804
  FALSE      1   1       0.9190153557  0.83719
28804
  FALSE      1   2       0.9190153557  0.83719
28804
  FALSE      1   3       0.9190153557  0.83719
28804
  FALSE      2   0       0.9190153557  0.83719
28804
  FALSE      2   1       0.9190153557  0.83719
28804
  FALSE      2   2       0.9190153557  0.83719
28804
  FALSE      2   3       0.9190153557  0.83719
28804
  FALSE      3   0       0.9190153557  0.83719
28804
```

```
  FALSE      3   1       0.9190153557  0.83719
28804
  FALSE      3   2       0.9190153557  0.83719
28804
  FALSE      3   3       0.9190153557  0.83719
28804
   TRUE      0   0                NaN
  NaN
   TRUE      0   1       0.9230120483  0.84521
98157
   TRUE      0   2       0.9120075742  0.82285
03015
   TRUE      0   3       0.9110053445  0.82076
06953
   TRUE      1   0                NaN
  NaN
   TRUE      1   1       0.9230120483  0.84521
98157
   TRUE      1   2       0.9120075742  0.82285
03015
   TRUE      1   3       0.9110053445  0.82076
06953
   TRUE      2   0                NaN
  NaN
   TRUE      2   1       0.9230120483  0.84521
98157
   TRUE      2   2       0.9120075742  0.82285
03015
   TRUE      2   3       0.9110053445  0.82076
06953
   TRUE      3   0                NaN
  NaN
   TRUE      3   1       0.9230120483  0.84521
98157
   TRUE      3   2       0.9120075742  0.82285
03015
   TRUE      3   3       0.9110053445  0.82076
```

```
06953

Accuracy was used to select the optimal model
using the largest value.
The final values used for the model were fL =
0, usekernel = TRUE and adjust
 = 1.
```

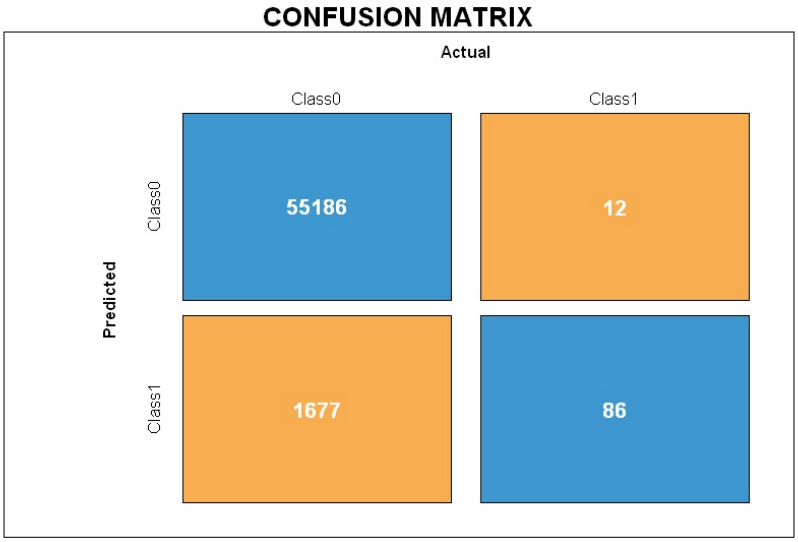
In [61]:

```r
#confusion matrix

pred_nb_cm = predict(nb_cm, newdata = fd_test)
cm15=confusionMatrix(table(pred_nb_cm,fd_test$Class))
cm15
draw_confusion_matrix(cm15)
x17=roc.curve(fd_test$Class, pred_nb_cm,curve = TRUE)
x17
plot(x17)
a17=(pr.curve(fd_test$Class, pred_nb_cm,curve = TRUE))
a17
plot(a17)
```

```
Confusion Matrix and Statistics


pred_nb_cm     0     1
         0 55186    12
         1  1677    86


               Accuracy : 0.9703481

                 95% CI : (0.9689222, 0.971725
9)
    No Information Rate : 0.9982795

    P-Value [Acc > NIR] : 1
```

```
                 Kappa : 0.0894553

Mcnemar's Test P-Value : <0.0000000000000002


            Sensitivity : 0.97050806

            Specificity : 0.87755102

         Pos Pred Value : 0.99978260

         Neg Pred Value : 0.04878049

             Prevalence : 0.99827952

         Detection Rate : 0.96883833

   Detection Prevalence : 0.96904900

      Balanced Accuracy : 0.92402954


       'Positive' Class : 0


 ROC curve

    Area under curve:
     0.4853847369
```

```
Curve for scores from  1  to  2
( can be plotted with plot(x) )
```

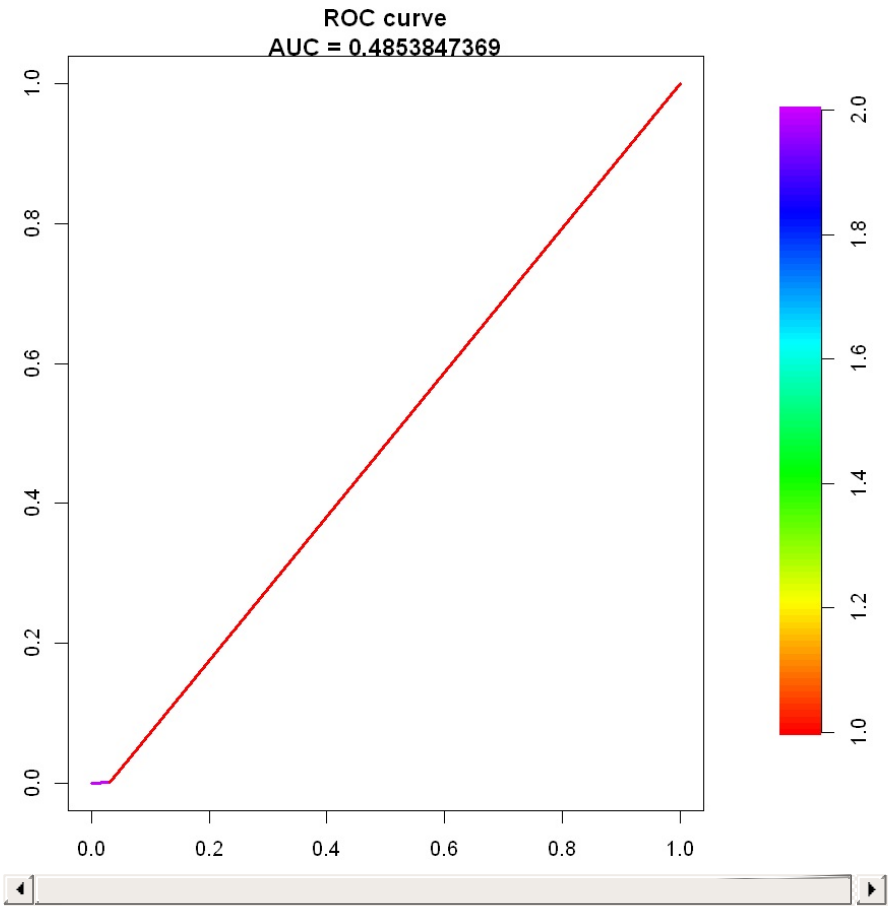


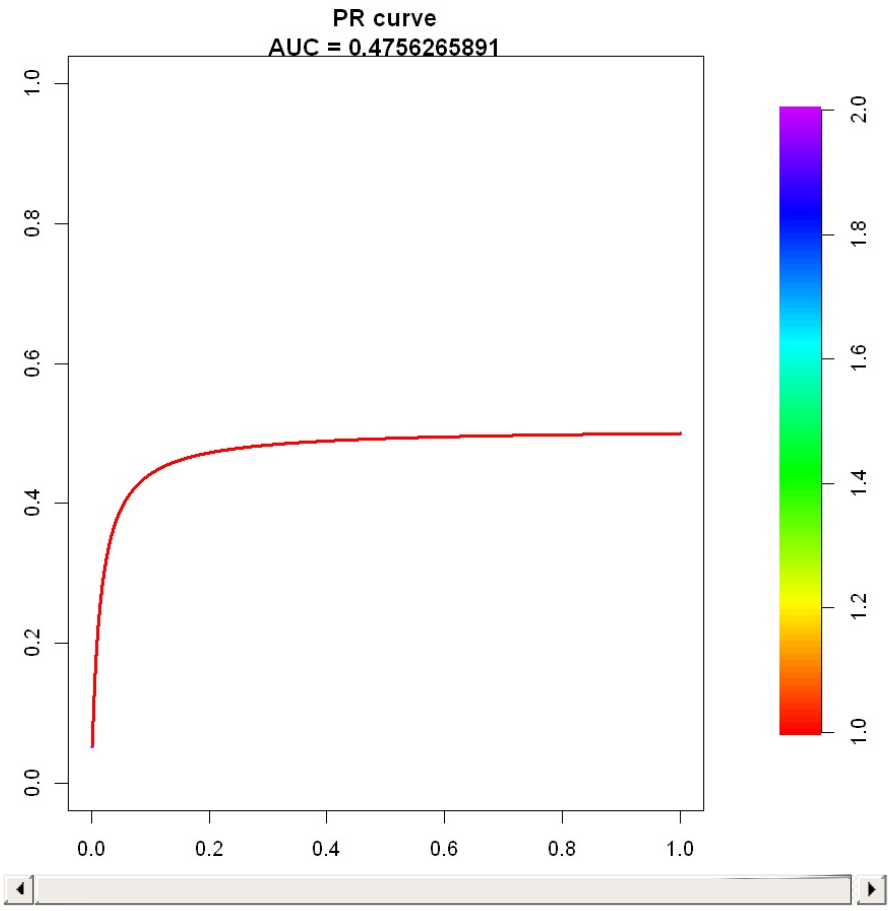
DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.971	0.878	1	0.971	0.985

Accuracy	Kappa
0.97	0.089

```
Precision-recall curve

  Area under curve (Integral):
   0.4756265891

  Area under curve (Davis & Goadrich):
   0.4756265883

  Curve for scores from  1  to  2
  ( can be plotted with plot(x) )
```


ROC curve
AUC = 0.4853847369
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0



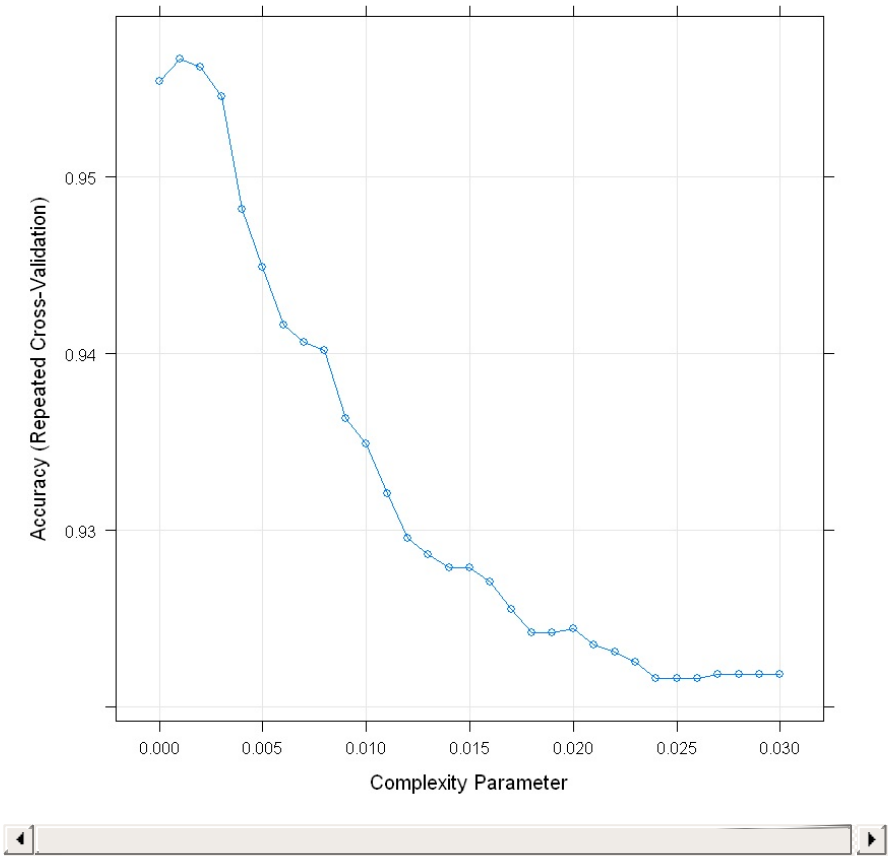
In [64]:

```
#Decision Tree
#Tuning
options(warn=-1)
dt_cm = train(Class~., data=fd_balanced_both, method='rpart',
              tuneGrid=expand.grid(.cp=seq(0.00,0.03,0.001)),
metric='Accuracy',
              trControl=trainControl(method='repeatedcv', num
ber=10,repeats=3))
dt_cm
plot(dt_cm)
```

```
CART

3000 samples
  30 predictor
```

```
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated
 3 times)
Summary of sample sizes: 2701, 2700, 2700, 270
0, 2699, 2699, ...
Resampling results across tuning parameters:

  cp     Accuracy      Kappa
  0.000  0.9554457235  0.9107724410
  0.001  0.9566646284  0.9131794093
  0.002  0.9562209247  0.9122973752
  0.003  0.9545505469  0.9089274356
  0.004  0.9482149740  0.8961222207
  0.005  0.9448823592  0.8894054334
  0.006  0.9416608555  0.8829396166
  0.007  0.9406608530  0.8809348441
  0.008  0.9402164086  0.8800385162
  0.009  0.9363267542  0.8721902285
  0.010  0.9348826715  0.8692779556
  0.011  0.9321093308  0.8636992091
  0.012  0.9295522889  0.8585489163
  0.013  0.9286630259  0.8567688273
  0.014  0.9278852481  0.8551986771
  0.015  0.9278852481  0.8551986771
  0.016  0.9271100543  0.8536322973
  0.017  0.9255533765  0.8504554000
  0.018  0.9242200432  0.8477208000
  0.019  0.9242200432  0.8477128290
  0.020  0.9244422654  0.8481531878
  0.021  0.9235504036  0.8463352801
  0.022  0.9231074358  0.8454382879
  0.023  0.9225518802  0.8443026951
  0.024  0.9216626197  0.8424853039
  0.025  0.9216626197  0.8424574464
  0.026  0.9216626197  0.8424502550
```

```
  0.027  0.9218848419  0.8428891764
  0.028  0.9218848419  0.8428891764
  0.029  0.9218848419  0.8428891764
  0.030  0.9218848419  0.8428891764

Accuracy was used to select the optimal model
using the largest value.
The final value used for the model was cp = 0.
001.
```



In [65]:

```
#Tuning2

fd_balanced_both_dt <- ovun.sample(Class ~ ., data = fd_train
_dt, method = "both",
                                   N = 1500, seed = 1)$data
table(fd_balanced_both_dt$Class)
```

```
tree_cm = rpart(Class ~ ., data =fd_balanced_both_dt,
                control=rpart.control(cp = 0.001,maxdepth = 8
,minsplit = 100))
tree_cm
```

```
  0   1
789 711

n= 1500

node), split, n, deviance, yval
      * denotes terminal node

 1) root 1500 373.986000000 0.474000000000
   2) V14>=-1.63201147 835  62.462275450 0.081
437125750
     4) V4< 2.440577663 767  36.117340290 0.04
9543676660
       8) V4< 0.7534115964 573   8.858638743 0
.015706806280
        16) V19< 1.028444536 528   2.982954545
 0.005681818182 *
        17) V19>=1.028444536 45   5.200000000
0.133333333300 *
       9) V4>=0.7534115964 194  24.664948450 0
.149484536100
        18) V12>=-0.5635721185 146   6.6643835
62 0.047945205480
          36) V4>=0.9340971258 113   0.0000000
00 0.000000000000 *
          37) V4< 0.9340971258 33   5.51515151
5 0.212121212100 *
        19) V12< -0.5635721185 48  11.91666667
0 0.458333333300 *
     5) V4>=2.440577663 68  16.764705880 0.441
176470600 *
   3) V14< -1.63201147 665  21.272180450 0.966
```

```
917293200
     6) V10>=-1.3248214 63  13.650793650 0.682
539682500 *
     7) V10< -1.3248214 602   1.993355482 0.99
6677740900 *
```

In [66]:

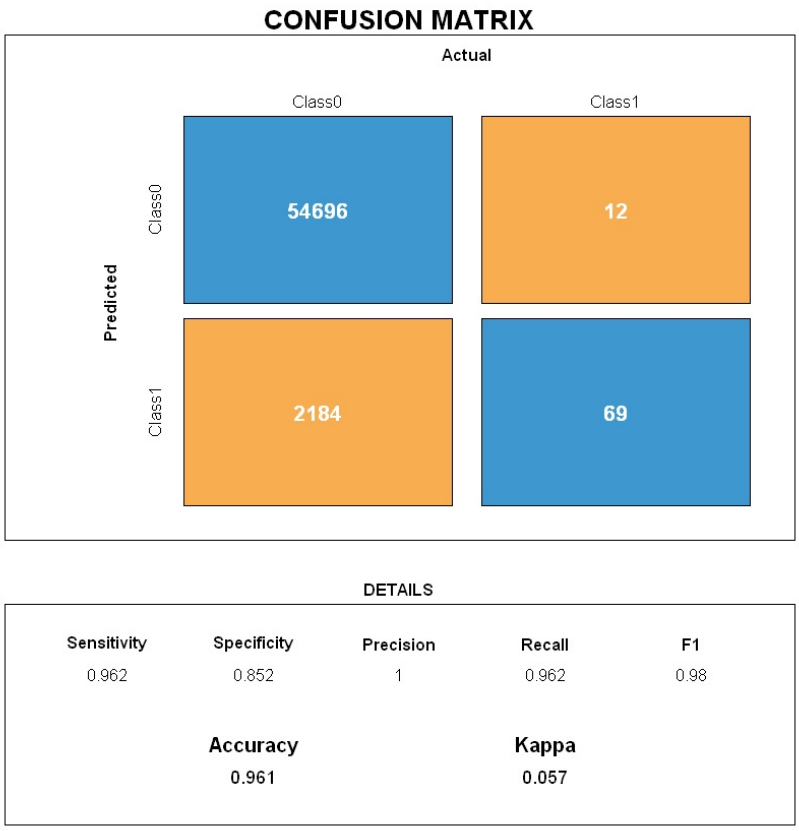
```
#confusion matrix

prune_cm <- prune(tree_cm, cp = 0.001)

pred_tree_cm <- predict(prune_cm, newdata = fd_test_dt)
pred_tree_cm_1 = ifelse(pred_tree_cm > 0.5, 1, 0)

cm16=confusionMatrix(table(pred_tree_cm_1,fd_test_dt$Class))
cm16
draw_confusion_matrix(cm16)
x18=roc.curve(fd_test_dt$Class, pred_tree_cm_1,curve=TRUE)
x18
plot(x18)
a18=(pr.curve(fd_test$Class, pred_tree_cm_1,curve = TRUE))
a18
plot(a18)
```

```
Confusion Matrix and Statistics


pred_tree_cm_1     0     1
             0 54696    12
             1  2184    69


               Accuracy : 0.9614473

                 95% CI : (0.9598338, 0.963013
6)
    No Information Rate : 0.998578
```

```
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0565358

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.96160338

            Specificity : 0.85185185

         Pos Pred Value : 0.99978065

         Neg Pred Value : 0.03062583

             Prevalence : 0.99857797

         Detection Rate : 0.96023595

   Detection Prevalence : 0.96044662

      Balanced Accuracy : 0.90672761

       'Positive' Class : 0


  ROC curve

    Area under curve:
```

0.4809343235

Curve for scores from 0 to 1
(can be plotted with plot(x))

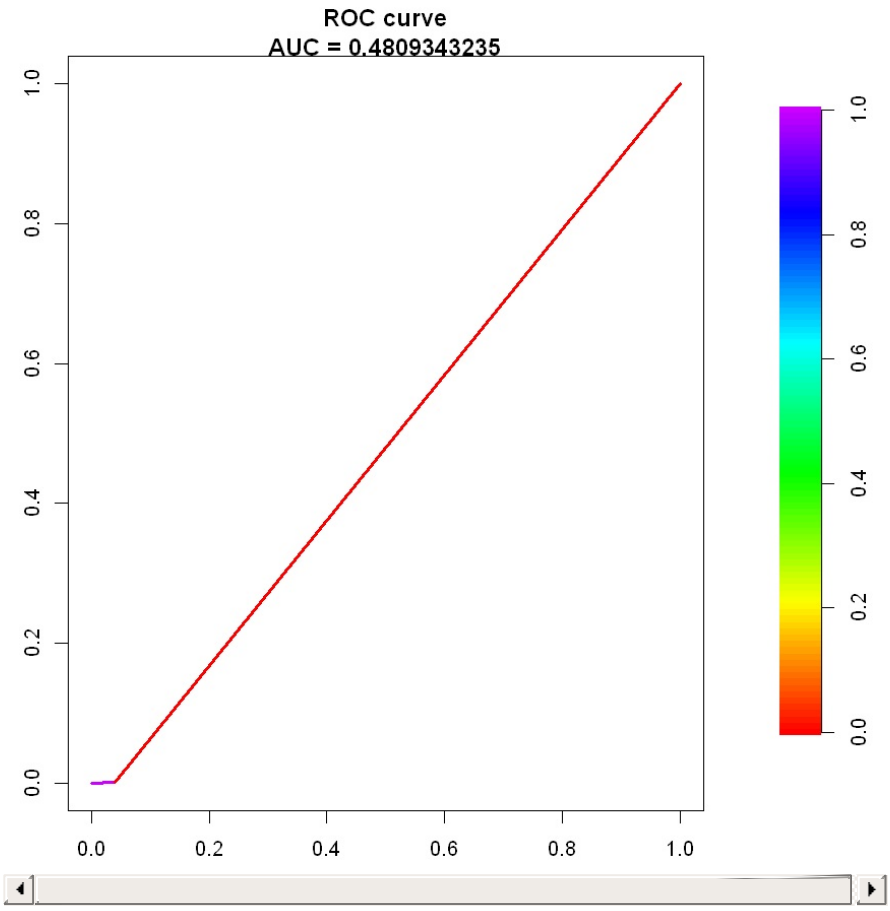


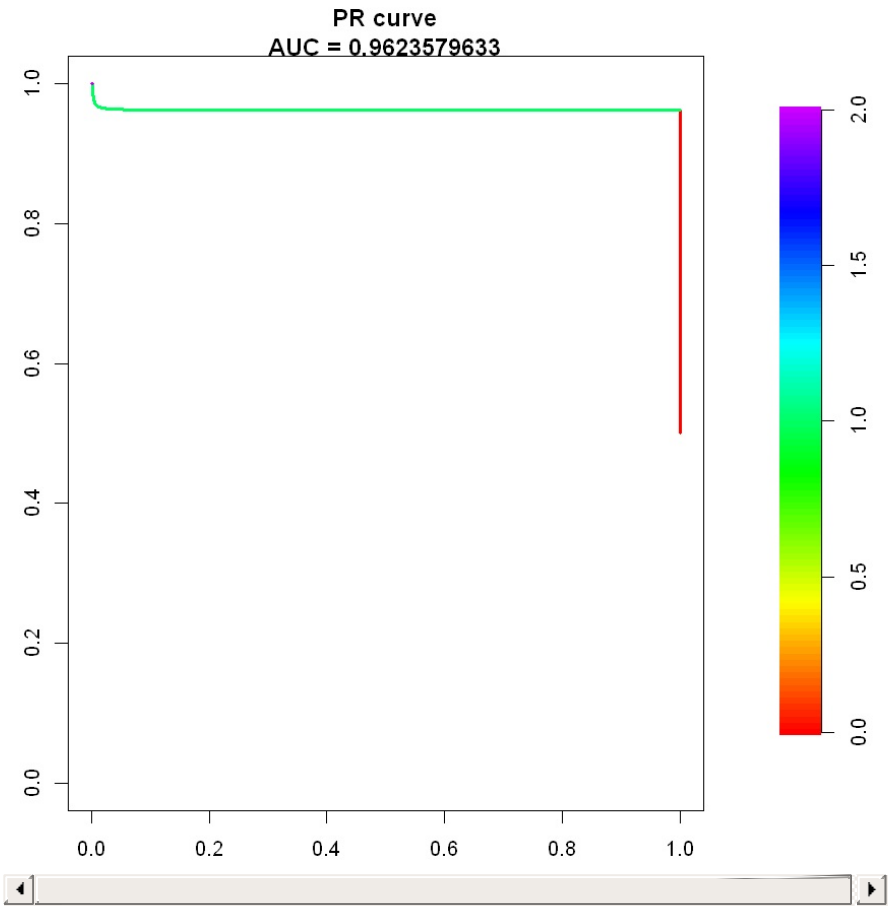
Precision-recall curve

Area under curve (Integral):
0.9623579633

Area under curve (Davis & Goadrich):
0.9623579639

Curve for scores from 0 to 2
(can be plotted with plot(x))


ROC curve
AUC = 0.4809343235
0.0
0.2
0.4
0.6
0.8
1.0

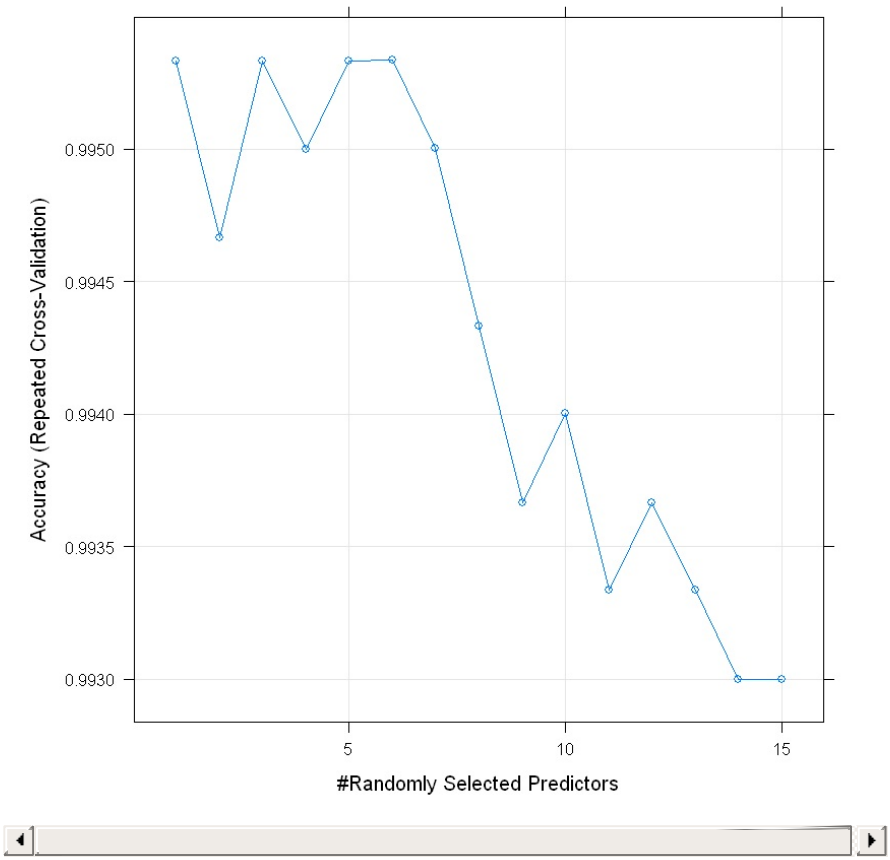


In [67]:

```
#Random forest
#tuning
options(warn=-1)
rf_cm = train(Class~., data=fd_balanced_both, method='rf',
               tuneGrid=expand.grid(.mtry=c(1:15)),
              metric='Accuracy',trControl=trainControl(method
='repeatedcv', number=10,repeats=1))
rf_cm
plot(rf_cm)
```

```
Random Forest

3000 samples
  30 predictor
   2 classes: '0', '1'
```

```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated
 1 times)
Summary of sample sizes: 2700, 2701, 2700, 270
0, 2701, 2700, ...
Resampling results across tuning parameters:

  mtry  Accuracy      Kappa
   1    0.9953310963  0.9906441101
   2    0.9946644296  0.9893089970
   3    0.9953310963  0.9906444669
   4    0.9949977629  0.9899769105
   5    0.9953322111  0.9906481999
   6    0.9953333259  0.9906513238
   7    0.9950011000  0.9899861452
   8    0.9943322111  0.9886476158
   9    0.9936666519  0.9873148803
  10    0.9940022074  0.9879865334
  11    0.9933344259  0.9866500253
  12    0.9936666519  0.9873148805
  13    0.9933355408  0.9866521330
  14    0.9930010926  0.9859828250
  15    0.9930010926  0.9859828250

Accuracy was used to select the optimal model
using the largest value.
The final value used for the model was mtry =
6.
```



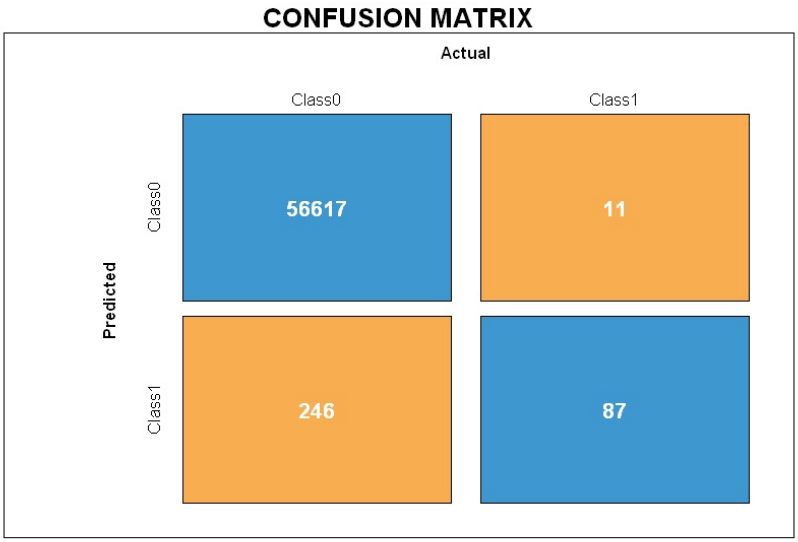
In [68]:

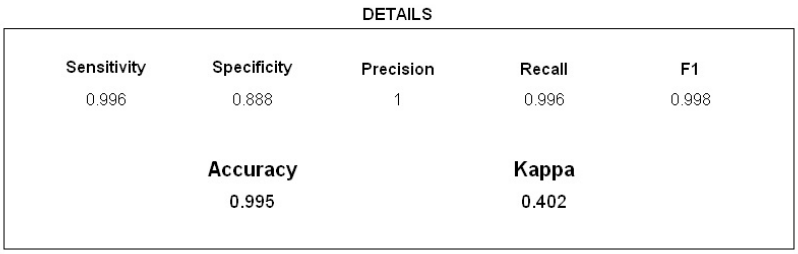
```
#confusion Matrix
options(warn=-1)
randomforest_cm = randomForest(x = fd_balanced_both[-29],
                               y = fd_balanced_both$Class,
                               ntree=2000,mtry = 6)

pred_rf_cm = predict(randomforest_cm, newdata = fd_test)
cm17=confusionMatrix(table(pred_rf_cm,fd_test$Class))
cm17
draw_confusion_matrix(cm17)
a=roc.curve(fd_test$Class, pred_rf_cm,curve = TRUE)
a
plot(a)
a19=(pr.curve(fd_test$Class, pred_rf_cm,curve = TRUE))
a19
```

```
plot(a19)
```

```
Confusion Matrix and Statistics


pred_rf_cm     0     1
         0 56617    11
         1   246    87


               Accuracy : 0.9954881

                 95% CI : (0.994903, 0.9960219
)
    No Information Rate : 0.9982795

    P-Value [Acc > NIR] : 1


                  Kappa : 0.4021228

 Mcnemar's Test P-Value : <0.0000000000000002


            Sensitivity : 0.9956738

            Specificity : 0.8877551

         Pos Pred Value : 0.9998057

         Neg Pred Value : 0.2612613

             Prevalence : 0.9982795

         Detection Rate : 0.9939608
```

```
 Detection Prevalence : 0.9941539

    Balanced Accuracy : 0.9417145


     'Positive' Class : 0


ROC curve

  Area under curve:
   0.4979371851

  Curve for scores from  1  to  2
  ( can be plotted with plot(x) )
```



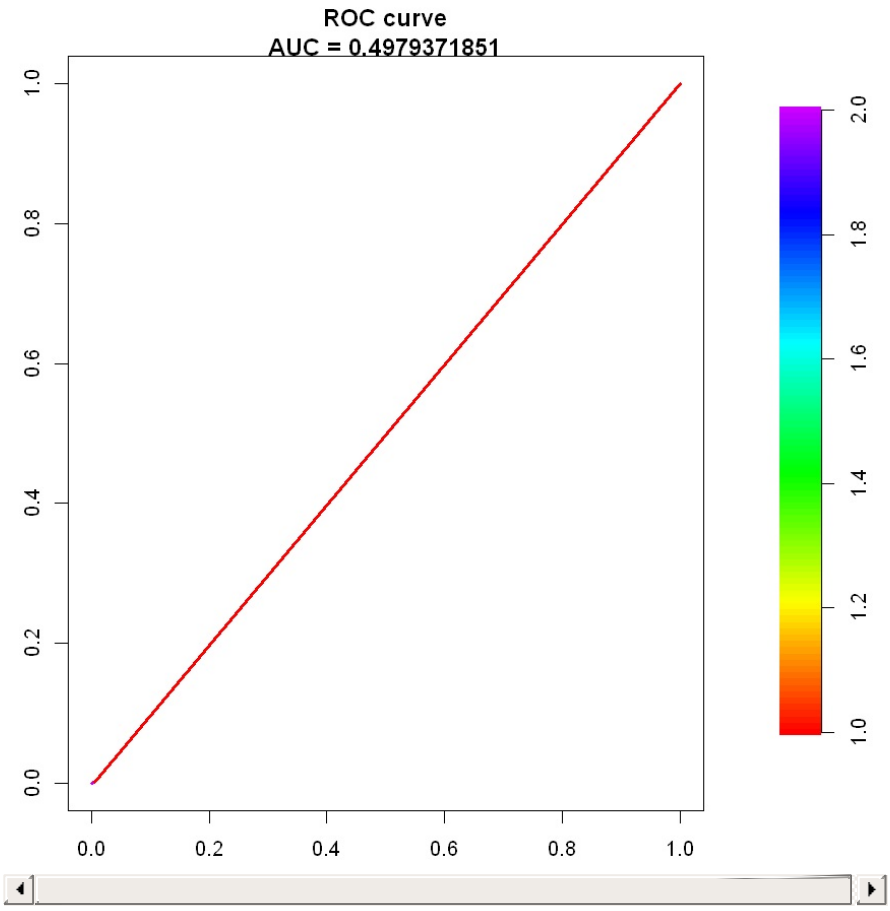


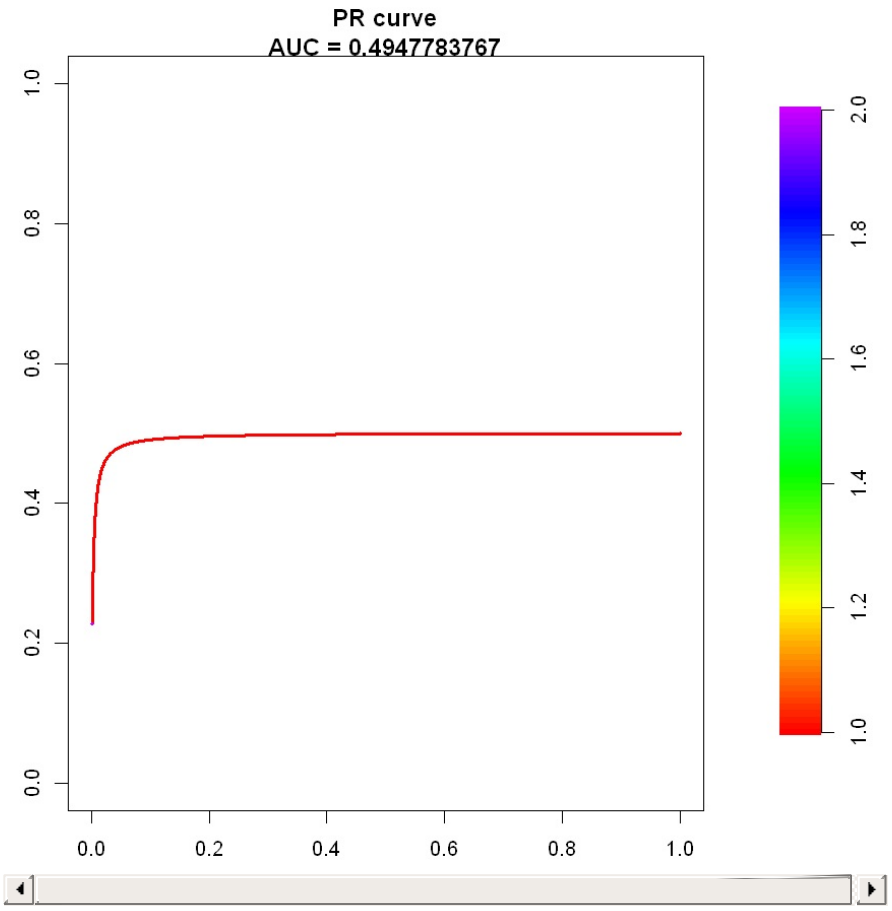
Precision-recall curve

Area under curve (Integral):
0.4947783767

Area under curve (Davis & Goadrich):
0.4947783749

Curve for scores from 1 to 2
(can be plotted with plot(x))


ROC curve
AUC = 0.4979371851
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0



In [69]:

```
#Support Vector Machine
#tuning
options(warn=-1)
svm_tn_os <- train(Class ~., data = fd_balanced_over,
                   method = "svmPoly",
                   trControl=trainControl(method = "repeatedcv",

                                          number = 10, repeats =
 1),
                   preProcess = c("center", "scale"),
                   tuneGrid = expand.grid(.degree = c(2:3),.scal
e = c(0.1,1,10),
                        .C = c(0,0.01, 0.05, 0.1, 0.5,1, 1.5, 2,5
)),
                   tuneLength = 10)
```

```
svm_tn_os
```

```
Support Vector Machines with Polynomial Kernel


1500 samples
  30 predictor
   2 classes: '0', '1'

Pre-processing: centered (30), scaled (30)
Resampling: Cross-Validated (10 fold, repeated
 1 times)
Summary of sample sizes: 1351, 1350, 1349, 135
0, 1351, 1350, ...
Resampling results across tuning parameters:

  degree  scale  C     Accuracy      Kappa

  2        0.1   0.00           NaN
NaN
  2        0.1   0.01  0.9212778049  0.8424355
701
  2        0.1   0.05  0.9392959391  0.8785112
135
  2        0.1   0.10  0.9486428138  0.8972292
757
  2        0.1   0.50  0.9613054210  0.9225851
465
  2        0.1   1.00  0.9633054210  0.9265924
584
  2        0.1   1.50  0.9646477029  0.9292817
992
  2        0.1   2.00  0.9633187845  0.9266254
140
  2        0.1   5.00  0.9633322666  0.9266624
951
  2        1.0   0.00           NaN
```

```
NaN
  2        1.0   0.01  0.9579587241  0.9158898
125
  2        1.0   0.05  0.9646879121  0.9293738
352
  2        1.0   0.10  0.9600211269  0.9200409
746
  2        1.0   0.50  0.9553277331  0.9106626
937
  2        1.0   1.00  0.9533499267  0.9067150
177
  2        1.0   1.50  0.9520210083  0.9040607
406
  2        1.0   2.00  0.9513587567  0.9027376
889
  2        1.0   5.00  0.9499987555  0.9000171
391
  2       10.0   0.00           NaN
NaN
  2       10.0   0.01  0.9460430834  0.8921054
812
  2       10.0   0.05  0.9420118820  0.8840505
242
  2       10.0   0.10  0.9400073485  0.8800378
282
  2       10.0   0.50  0.9400073485  0.8800378
282
  2       10.0   1.00  0.9400073485  0.8800378
282
  2       10.0   1.50  0.9400073485  0.8800378
282
  2       10.0   2.00  0.9400073485  0.8800378
282
  2       10.0   5.00  0.9400073485  0.8800378
282
  3        0.1   0.00           NaN
NaN
```

```
  3         0.1   0.01  0.9332913463  0.8664875
249
  3         0.1   0.05  0.9526339837  0.9052266
467
  3         0.1   0.10  0.9586342208  0.9172390
818
  3         0.1   0.50  0.9659988740  0.9319881
415
  3         0.1   1.00  0.9666745485  0.9333448
072
  3         0.1   1.50  0.9646789635  0.9293549
070
  3         0.1   2.00  0.9666834970  0.9333645
398
  3         0.1   5.00  0.9613411559  0.9226858
832
  3         1.0   0.00           NaN
NaN
  3         1.0   0.01  0.9646966828  0.9294006
559
  3         1.0   0.05  0.9613322666  0.9226744
453
  3         1.0   0.10  0.9633279109  0.9266627
512
  3         1.0   0.50  0.9633279109  0.9266627
512
  3         1.0   1.00  0.9633279109  0.9266627
512
  3         1.0   1.50  0.9633279109  0.9266627
512
  3         1.0   2.00  0.9633279109  0.9266627
512
  3         1.0   5.00  0.9633279109  0.9266627
512
  3        10.0   0.00           NaN
NaN
  3        10.0   0.01  0.9646613035  0.9293258
```

```
955
  3       10.0   0.05  0.9646613035  0.9293258
955
  3       10.0   0.10  0.9646613035  0.9293258
955
  3       10.0   0.50  0.9646613035  0.9293258
955
  3       10.0   1.00  0.9646613035  0.9293258
955
  3       10.0   1.50  0.9646613035  0.9293258
955
  3       10.0   2.00  0.9646613035  0.9293258
955
  3       10.0   5.00  0.9646613035  0.9293258
955

Accuracy was used to select the optimal model
using the largest value.
The final values used for the model were degre
e = 3, scale = 0.1 and C = 2.
```

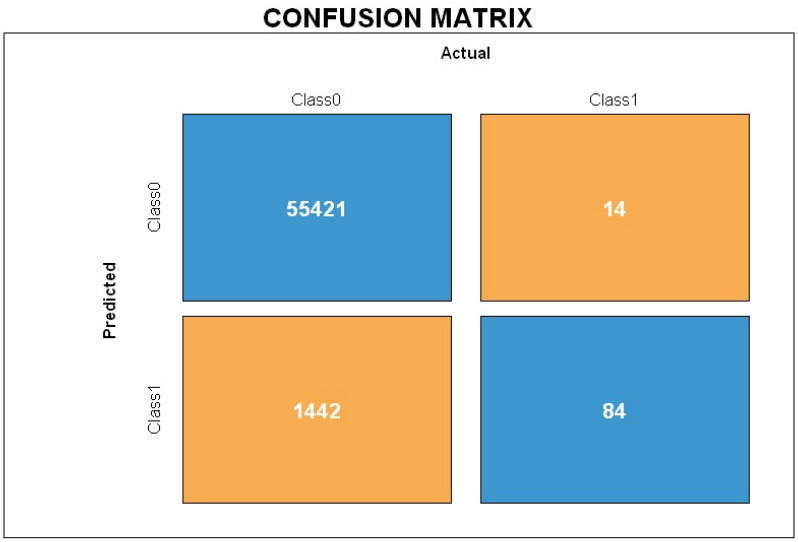
In [70]:

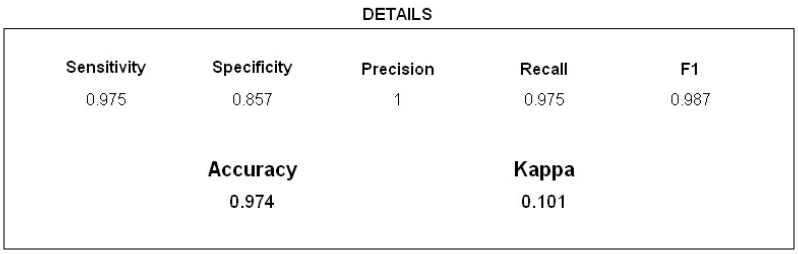
```
#tuning and confusion matrix
options(warn=-1)
tune_out_cm = tune.svm(x = fd_balanced_both[,-29], y = fd_bal
anced_both[,29],
                       type = "C-classification", kernel = "p
olynomial", degree = 3,
                       cost = 2, gamma = c(0.1,1,10), coef0 =
 c(0.1,1,10))

svm_cm = svm(Class~ ., data = fd_balanced_both, type = "C-cla
ssification",
             kernel = "polynomial", degree = 3,scale =0.1,
             cost = tune_out_cm$best.parameters$cost,
             gamma = tune_out_cm$best.parameters$gamma,
```

```
            coef0 = tune_out_cm$best.parameters$coef0)

pred_svm_cm = predict(svm_cm, newdata = fd_test)
cm18=confusionMatrix(table(pred_svm_cm,fd_test$Class))
cm18
draw_confusion_matrix(cm18)
b=roc.curve(fd_test$Class, pred_svm_cm,curve = TRUE)
b
plot(b)
a20=(pr.curve(fd_test$Class, pred_svm_cm,curve = TRUE))
a20
plot(a20)
```

```
Confusion Matrix and Statistics


pred_svm_cm     0     1
          0 55421    14
          1  1442    84


               Accuracy : 0.9744387

                 95% CI : (0.9731094, 0.975719
3)
    No Information Rate : 0.9982795

    P-Value [Acc > NIR] : 1


                  Kappa : 0.10054

 Mcnemar's Test P-Value : <0.0000000000000002
```

```
            Sensitivity : 0.97464080
            Specificity : 0.85714286
         Pos Pred Value : 0.99974745
         Neg Pred Value : 0.05504587
             Prevalence : 0.99827952
         Detection Rate : 0.97296396
   Detection Prevalence : 0.97320974
      Balanced Accuracy : 0.91589183

       'Positive' Class : 0


ROC curve

  Area under curve:
   0.4874651077

  Curve for scores from  1  to  2
  ( can be plotted with plot(x) )
```



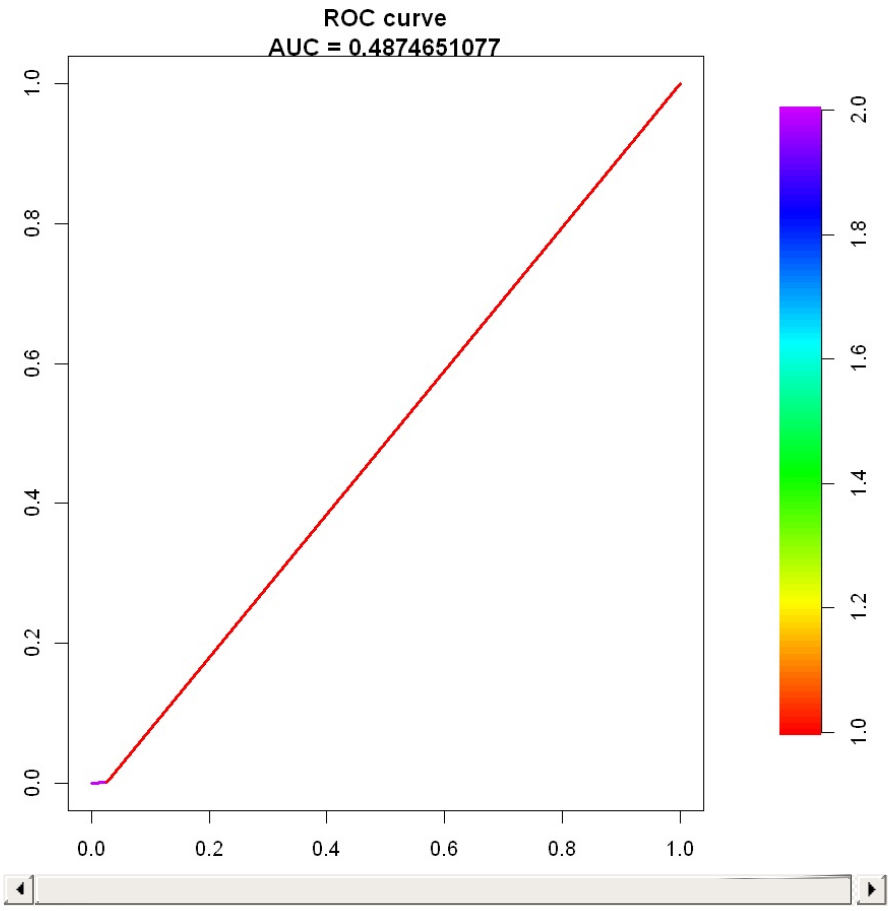


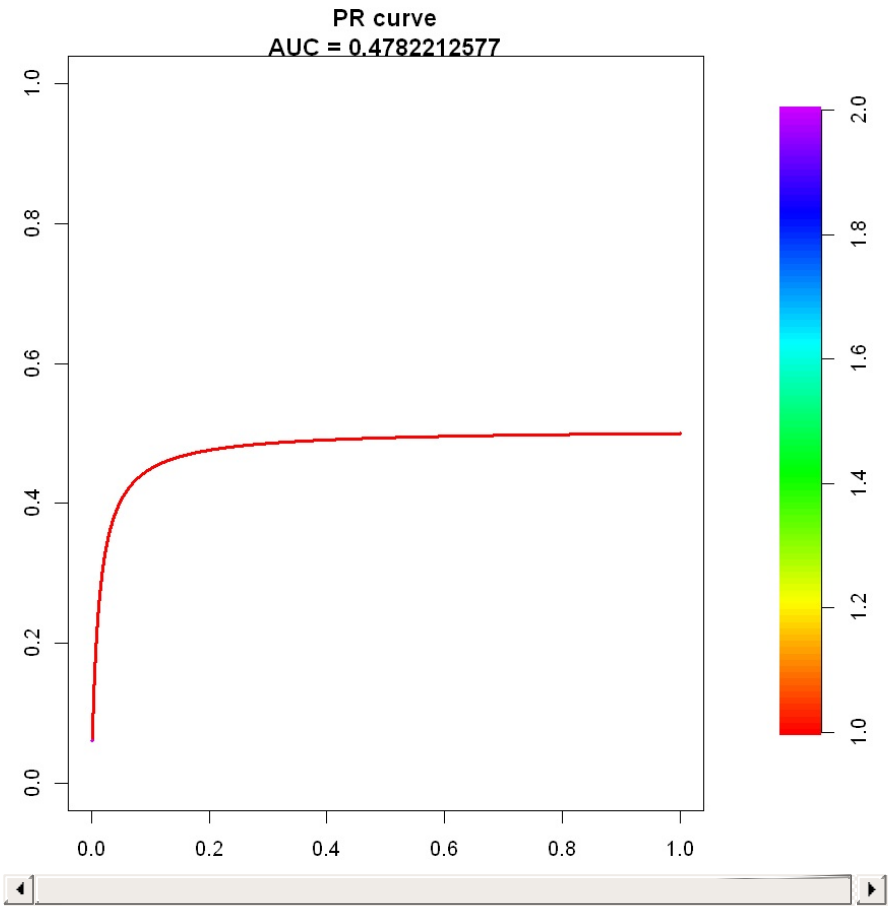
Precision-recall curve

Area under curve (Integral):
0.4782212577

Area under curve (Davis & Goadrich):
0.4782212569

Curve for scores from 1 to 2
(can be plotted with plot(x))


ROC curve
AUC = 0.4874651077
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0

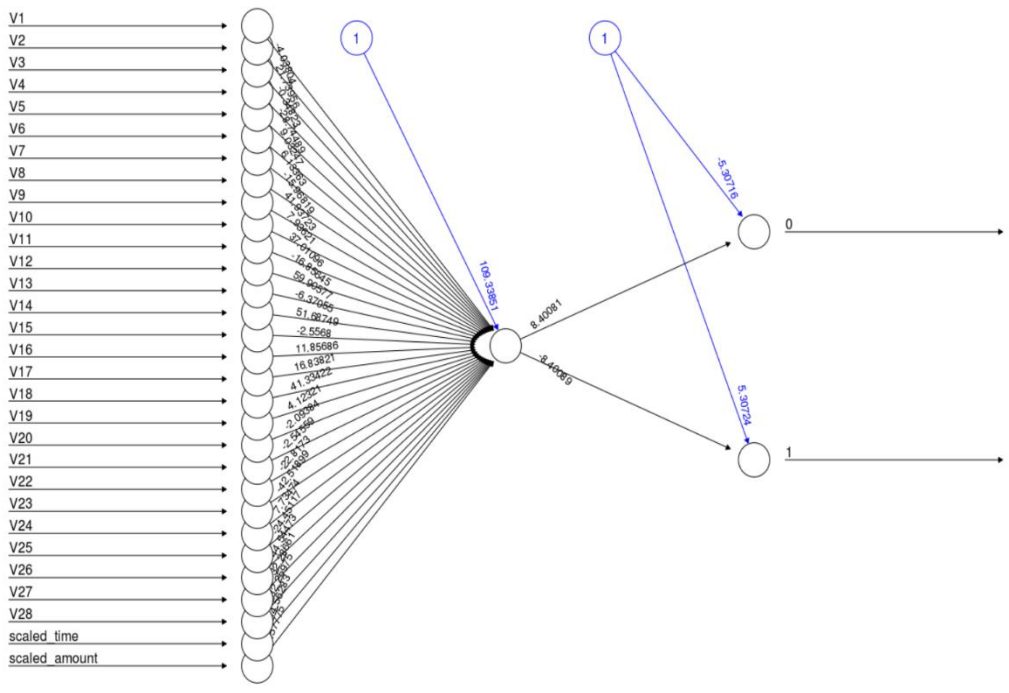

PR curve
AUC = 0.4782212577
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0

#Neural Network

```{r neural net library,warning=FALSE}

library(neuralnet)

library(GGally)

```

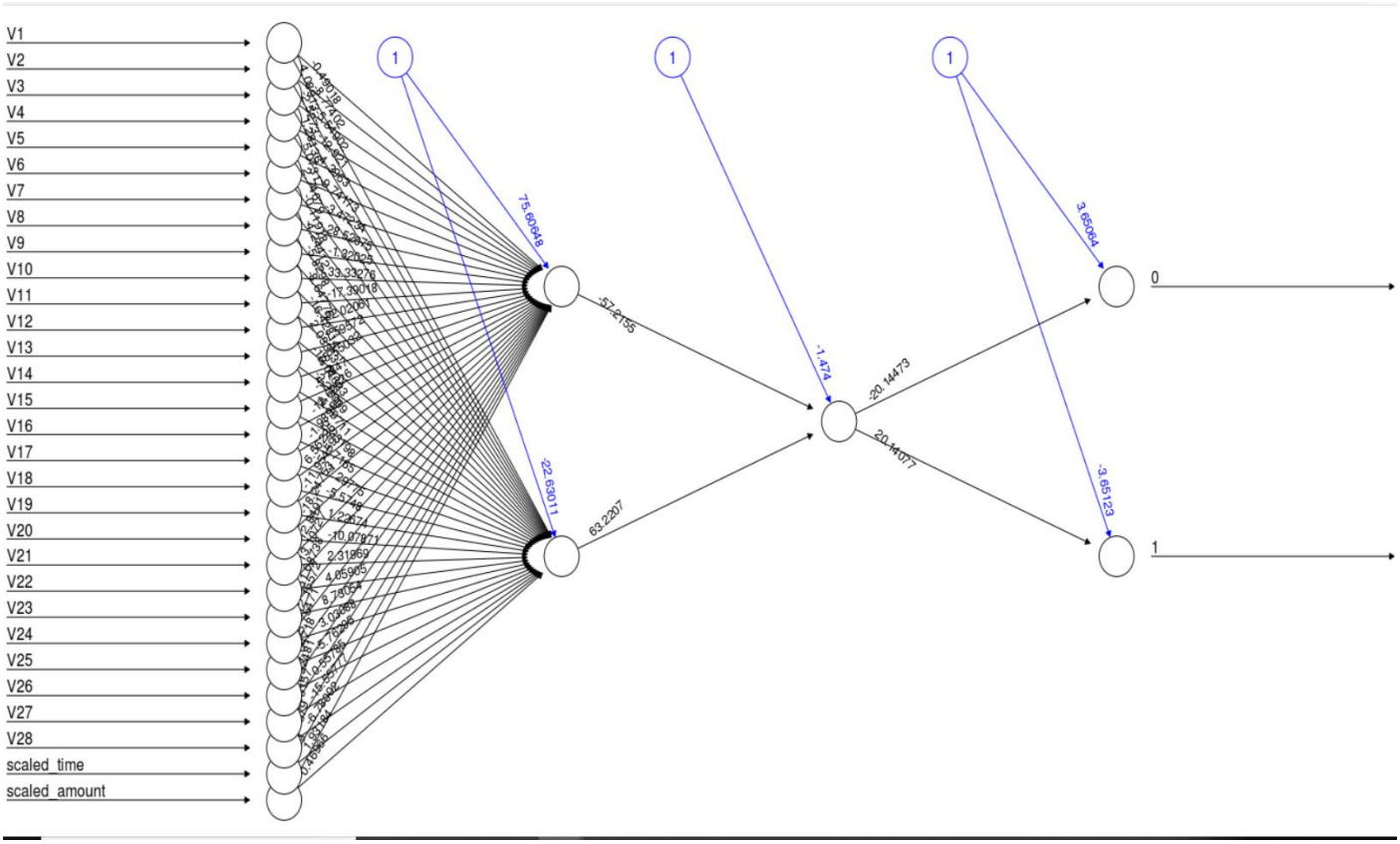
#undersampling

```{r NN US,warning=FALSE}

fd_NN1 = neuralnet(Class~.,

                   data = fd_balanced_under,

                   linear.output = FALSE,

                   err.fct = 'ce',

                   likelihood = TRUE)

plot(fd_NN1, rep = 'best')

```

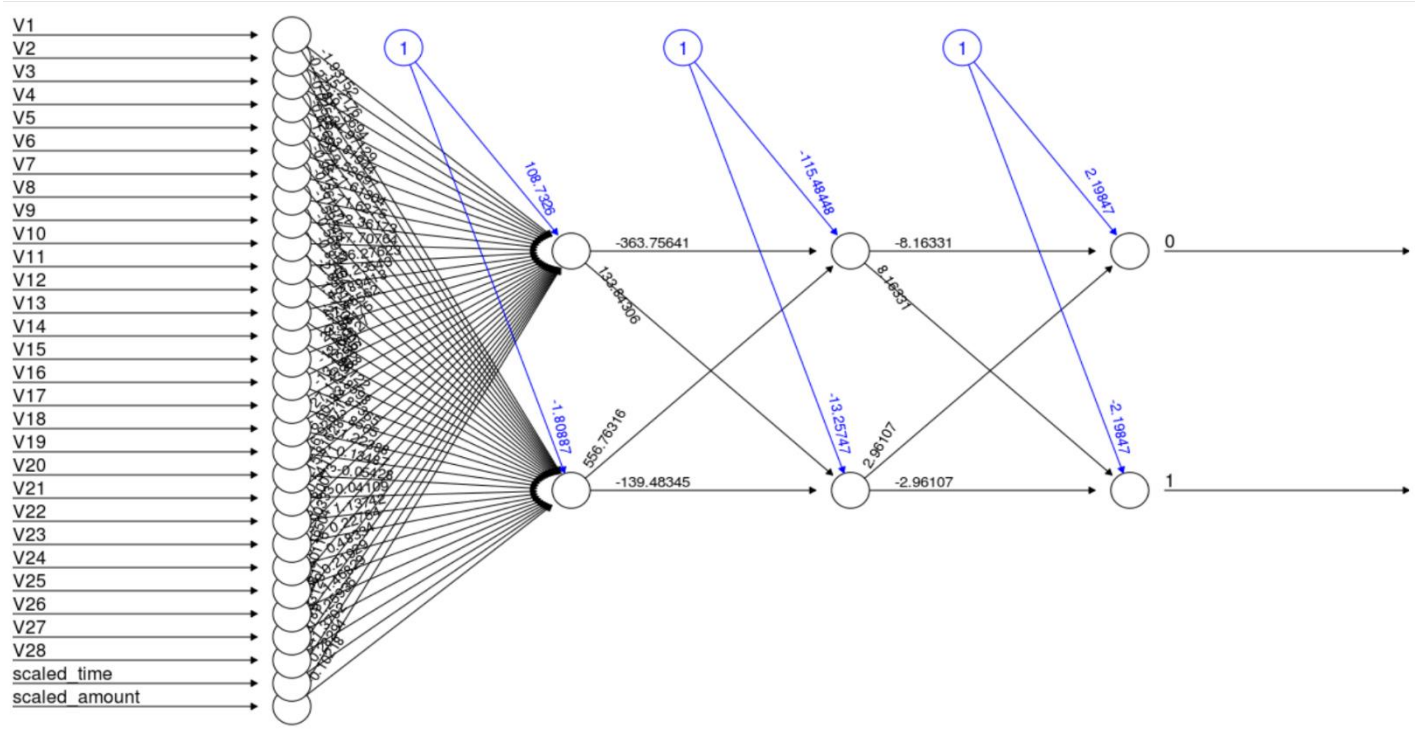



```{r nn2,warning=FALSE}

fd_NN2 <- neuralnet(Class~.,

                   data = fd_balanced_under,
```

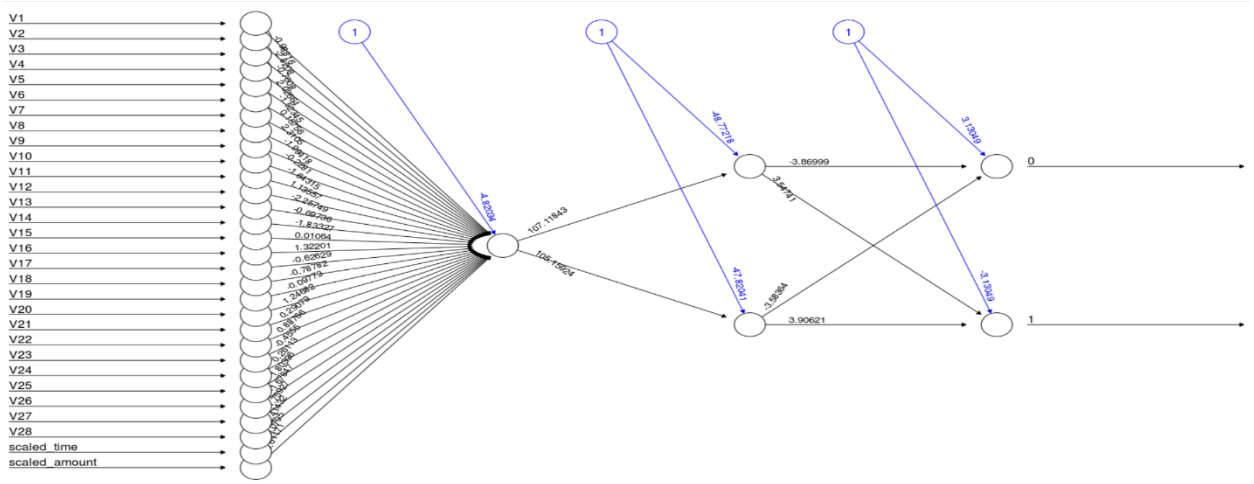
```
                linear.output = FALSE,
                err.fct = 'ce',
                likelihood = TRUE,hidden = c(2,1))
plot(fd_NN2, rep = 'best')
```




```{r nn3}
fd_NN3 <- neuralnet(Class~.,
                data = fd_balanced_under,
                linear.output = FALSE,
                err.fct = 'ce',
                likelihood = TRUE,hidden = c(2,2))
plot(fd_NN3, rep = 'best')
```

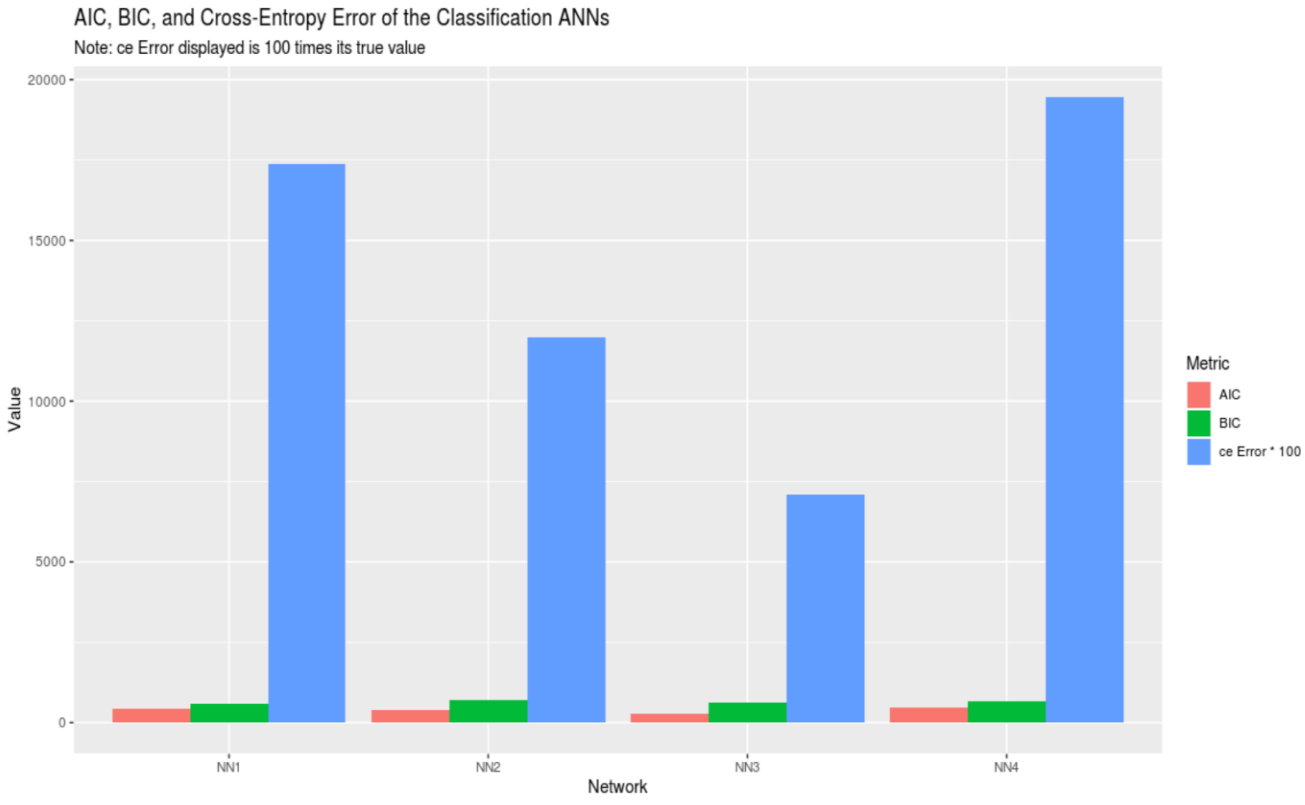



```{r nn4}

fd_NN4 <- neuralnet(Class~.,

data = fd_balanced_under,

linear.output = FALSE,

err.fct = 'ce',

likelihood = TRUE,hidden = c(1,2))

plot(fd_NN4, rep = 'best')

```

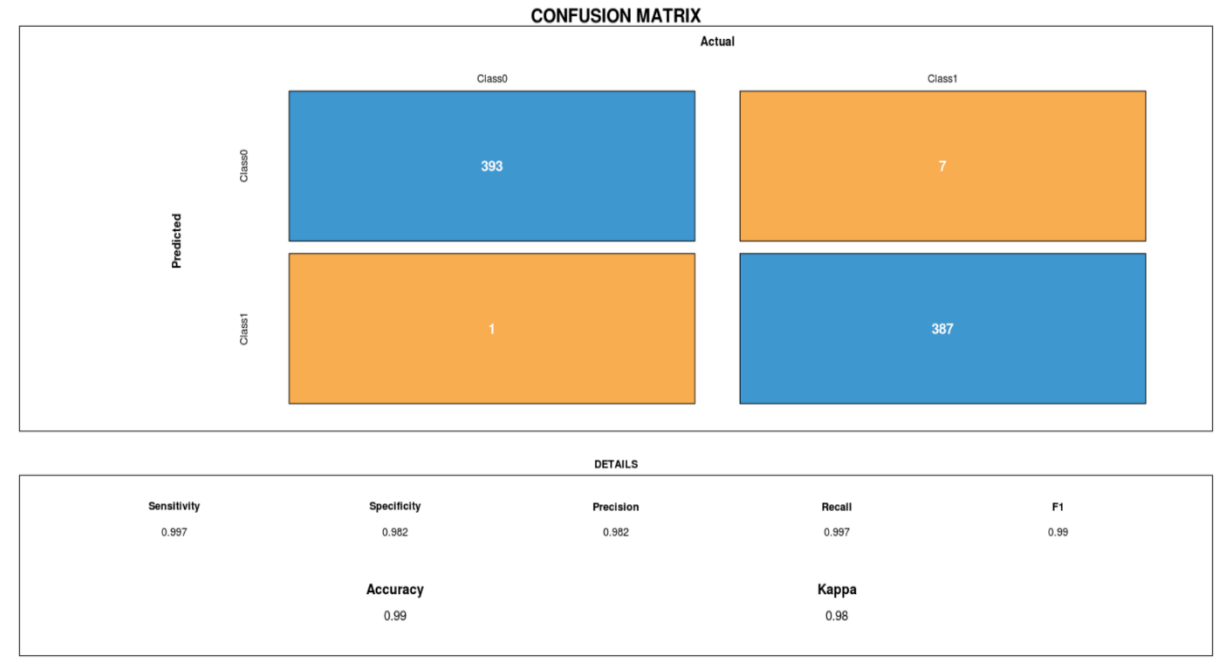


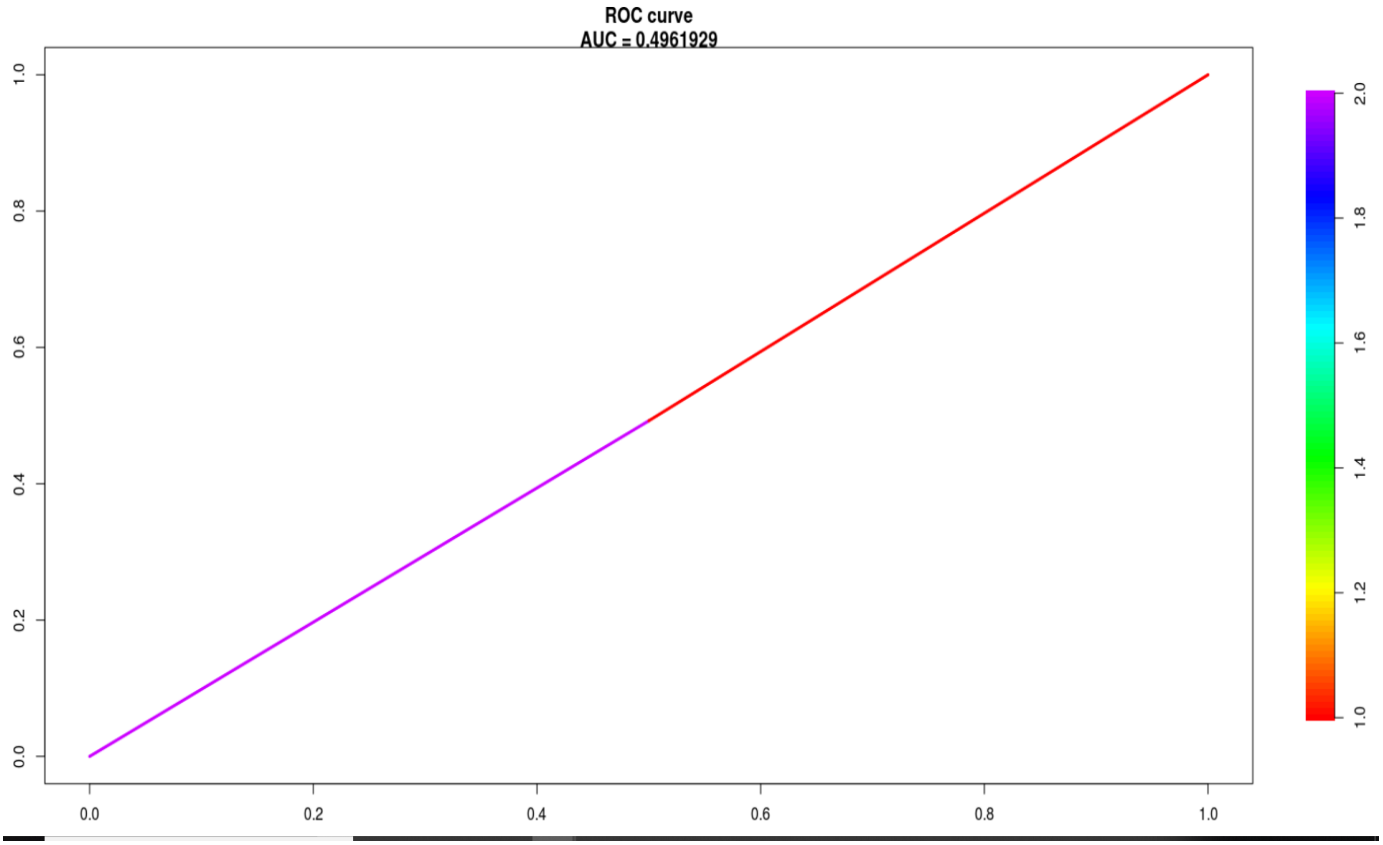
#NN AIC

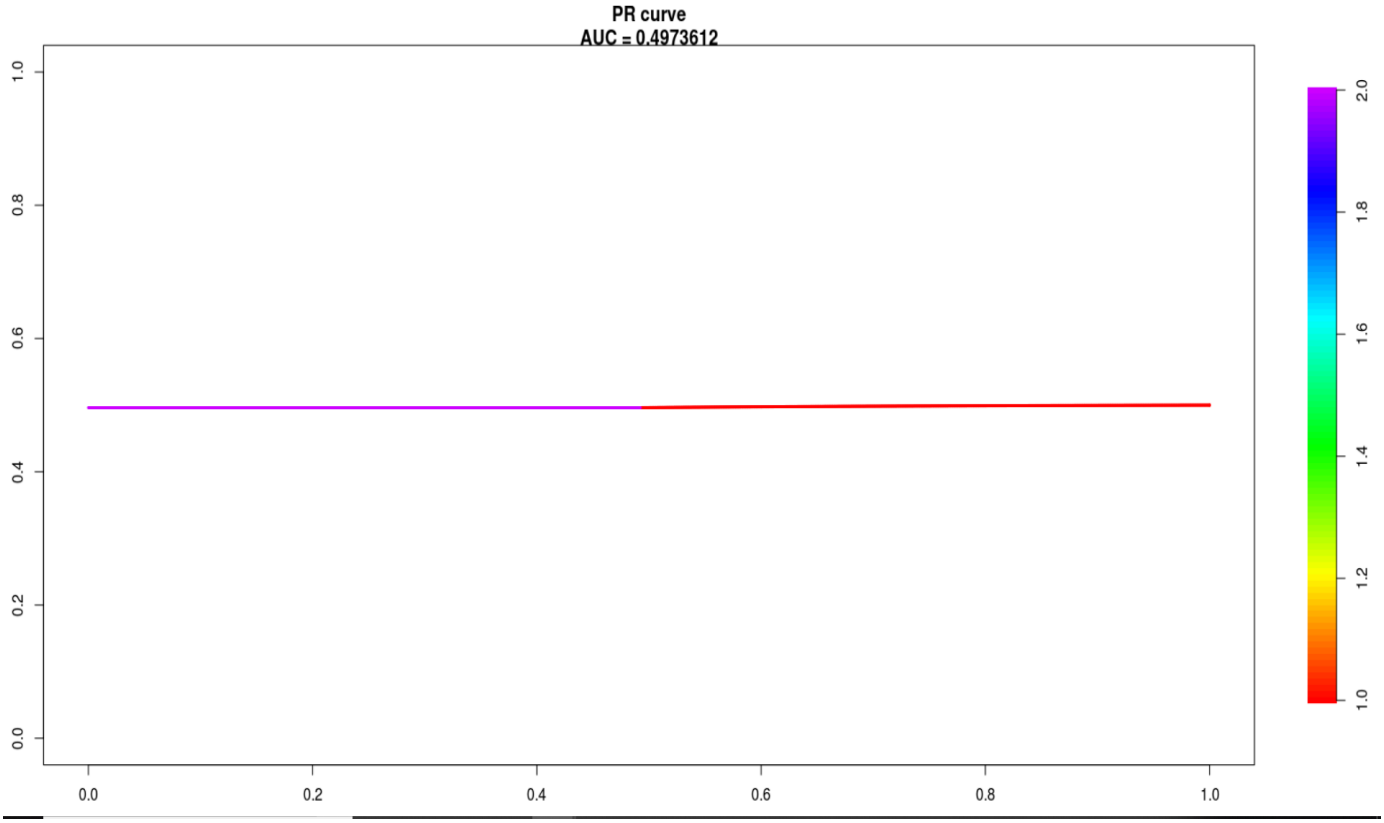
```{r nn aic}

Class_NN_ICs <- tibble('Network' = rep(c("NN1", "NN2", "NN3", "NN4"), each = 3),

'Metric' = rep(c('AIC', 'BIC', 'ce Error * 100'), length.out=12),'Value' =c(fd_NN1$result.matrix[4,1],fd_NN1$result.matrix[5,1], 100*fd_NN1$result.matrix[1,1], fd_NN2$result.matrix[4,1],

fd_NN2$result.matrix[5,1], 100*fd_NN2$result.matrix[1,1],

fd_NN3$result.matrix[4,1],fd_NN3$result.matrix[5,1],

100*fd_NN3$result.matrix[1,1], fd_NN4$result.matrix[4,1],

fd_NN4$result.matrix[5,1],100*fd_NN4$result.matrix[1,1]))

Class_NN_ICs %>%

ggplot(aes(Network, Value, fill = Metric)) +

geom_col(position = 'dodge') +

ggtitle("AIC, BIC, and Cross-Entropy Error of the Classification ANNs", "Note: ce Error displayed is 100 times its true value")

```



```{r NN conmat}
predict <- predict(fd_NN3, fd_balanced_under[,-29])
predicted.class <- apply(predict,1,which.max)-1

cm19=confusionMatrix(factor(ifelse(predicted.class == "0", "0", "1")),
        factor(fd_balanced_under$Class))
draw_confusion_matrix(cm19)
x7=roc.curve(factor(ifelse(predicted.class == "0", "0", "1")),
        factor(fd_balanced_under$Class),curve = TRUE)
x7
plot(x7)
#recall
a7=(pr.curve(factor(ifelse(predicted.class == "0", "0", "1")),
        factor(fd_balanced_under$Class),curve = TRUE))
a7
plot(a7)
```



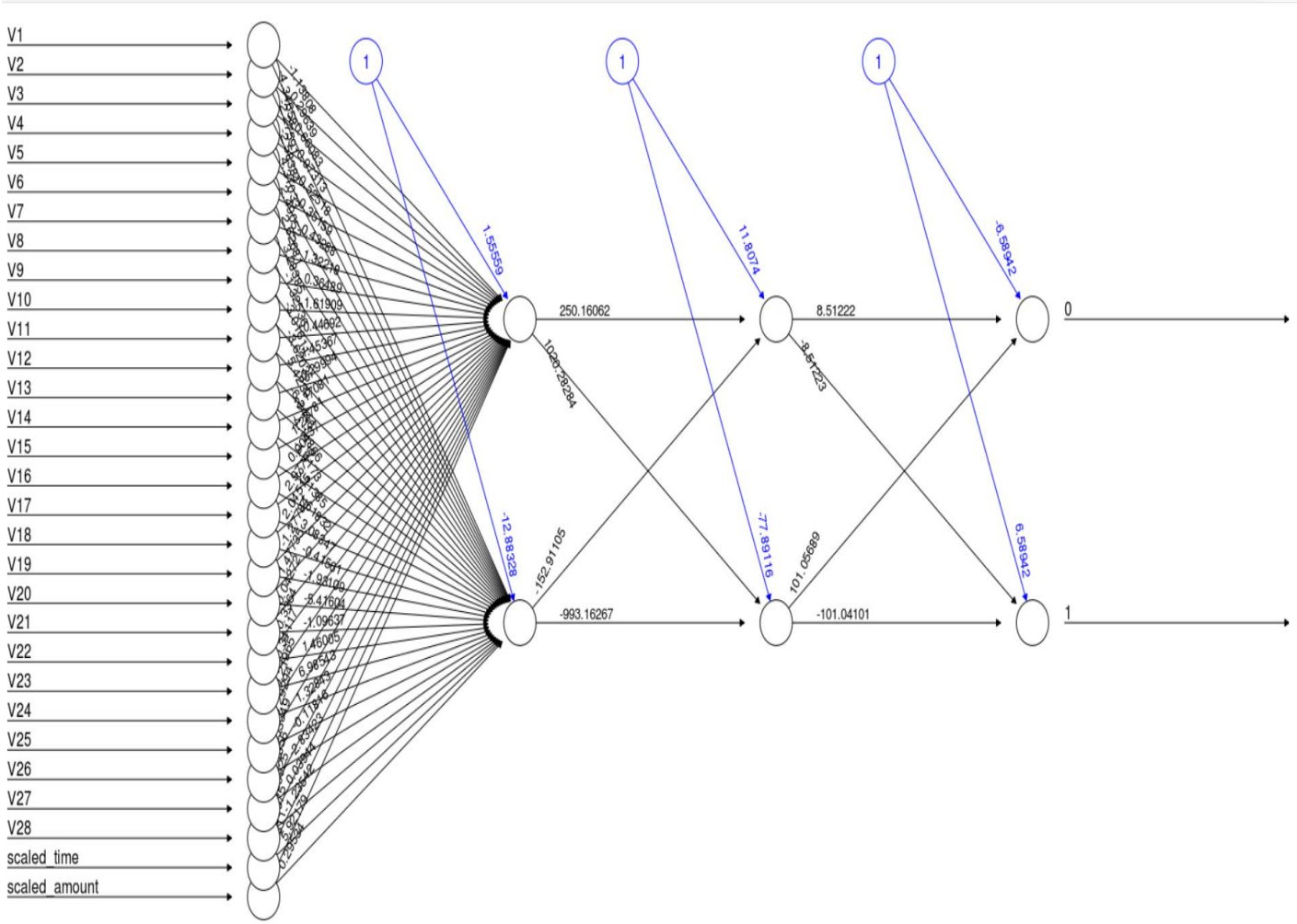

ROC curve
AUC = 0.4961929
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0


PR curve
AUC = 0.4973612
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0

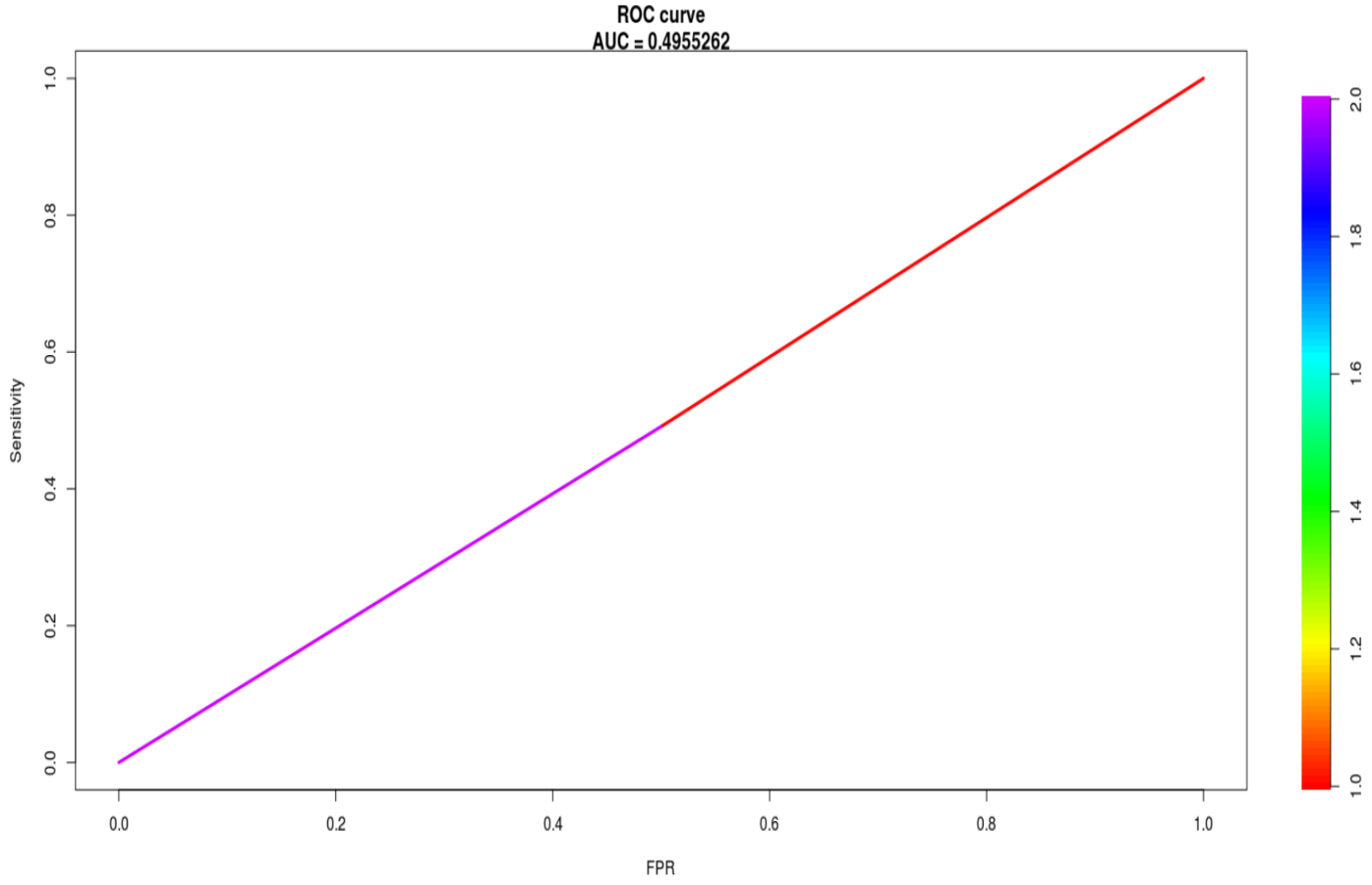
#oversampling:

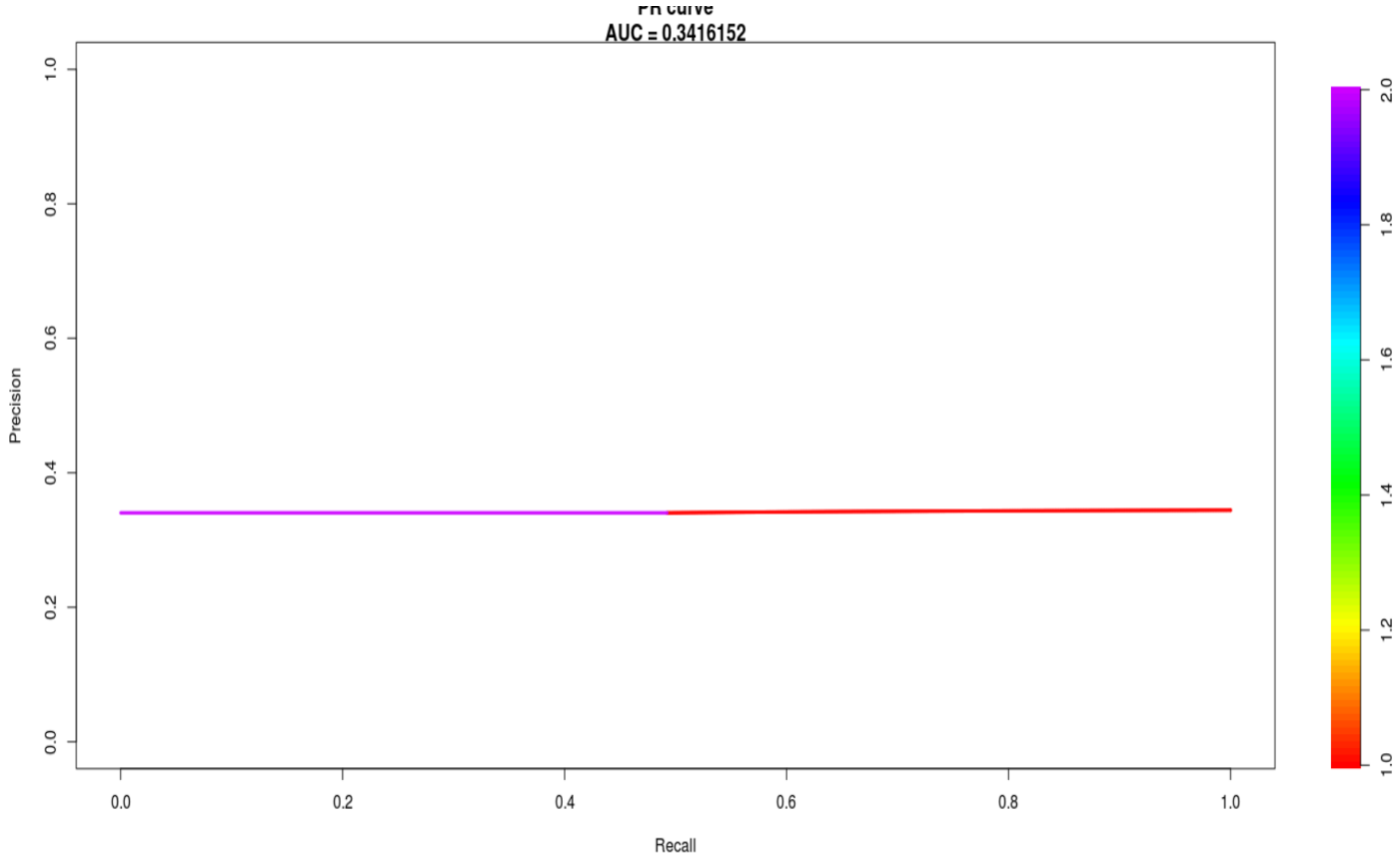
#ANN

```{r ANN OS,warning=TRUE}
fd_NN3_o <- neuralnet(Class~.,
                      data = fd_balanced_over,
                      linear.output = FALSE,
                      err.fct = 'ce',
                      likelihood = TRUE,hidden = c(2,2))
plot(fd_NN3_o, rep = 'best')
```



```{r ANN os cm}
predict_o <- predict(fd_NN3_o, fd_balanced_over[,-29])
predicted.class_o <- apply(predict,1,which.max)-1
x14=roc.curve(factor(ifelse(predicted.class_o == "0", "0", "1")),
              factor(fd_balanced_over$Class),curve = TRUE)
x14
plot(x14)
#recall
a14=(pr.curve(factor(ifelse(predicted.class_o == "0", "0", "1")),
              factor(fd_balanced_over$Class),curve = TRUE))
a14
plot(a14)
```

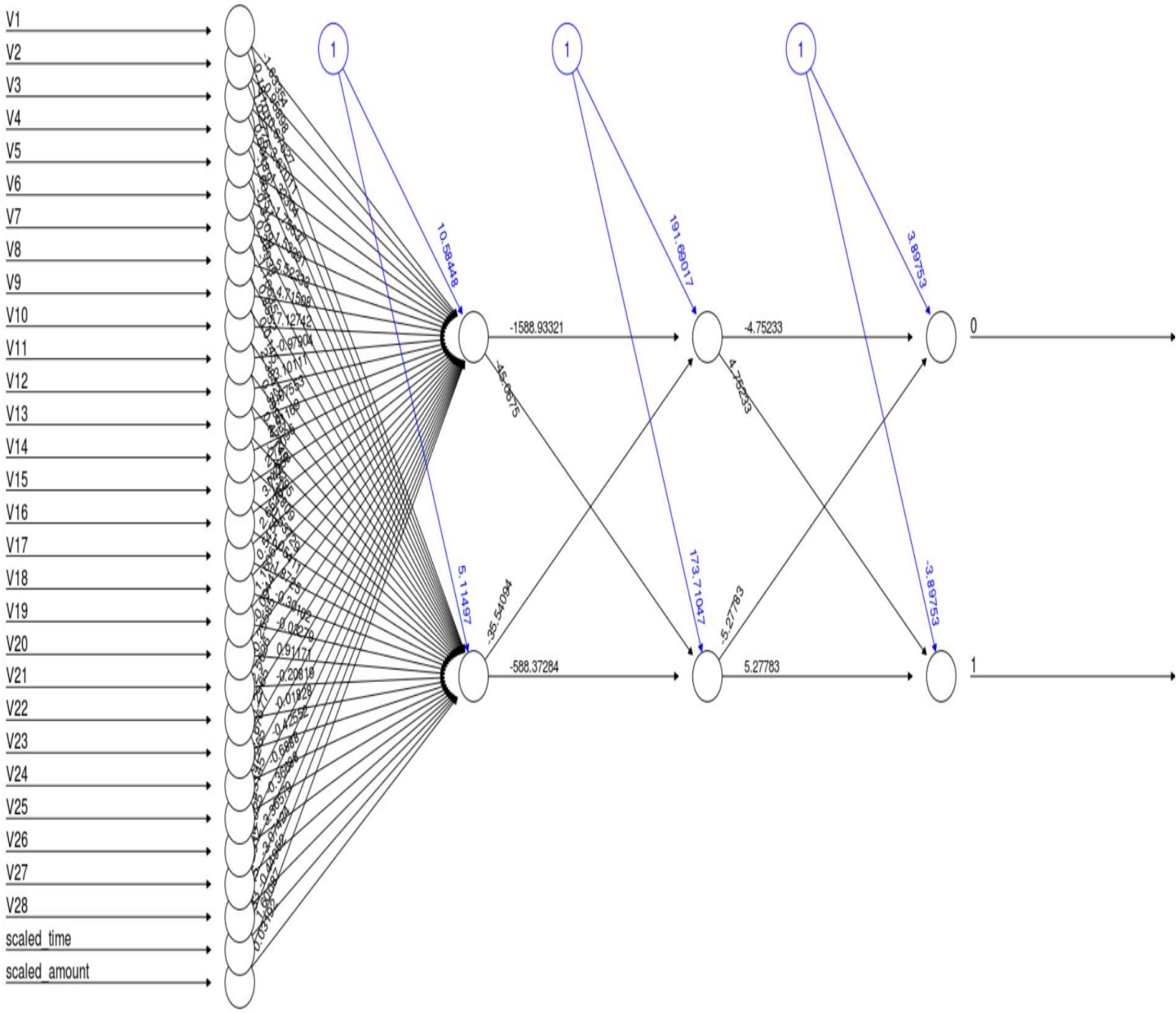


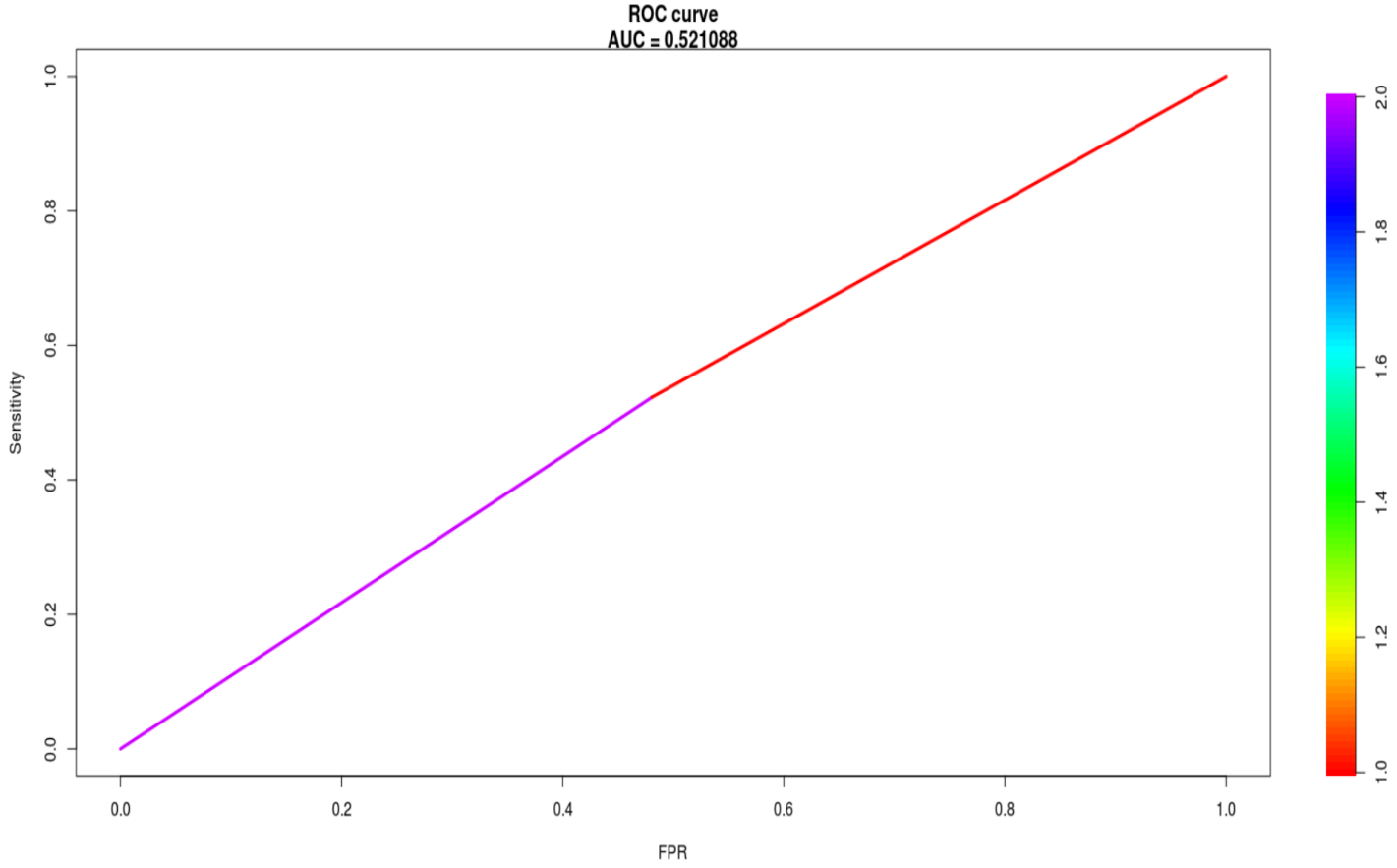


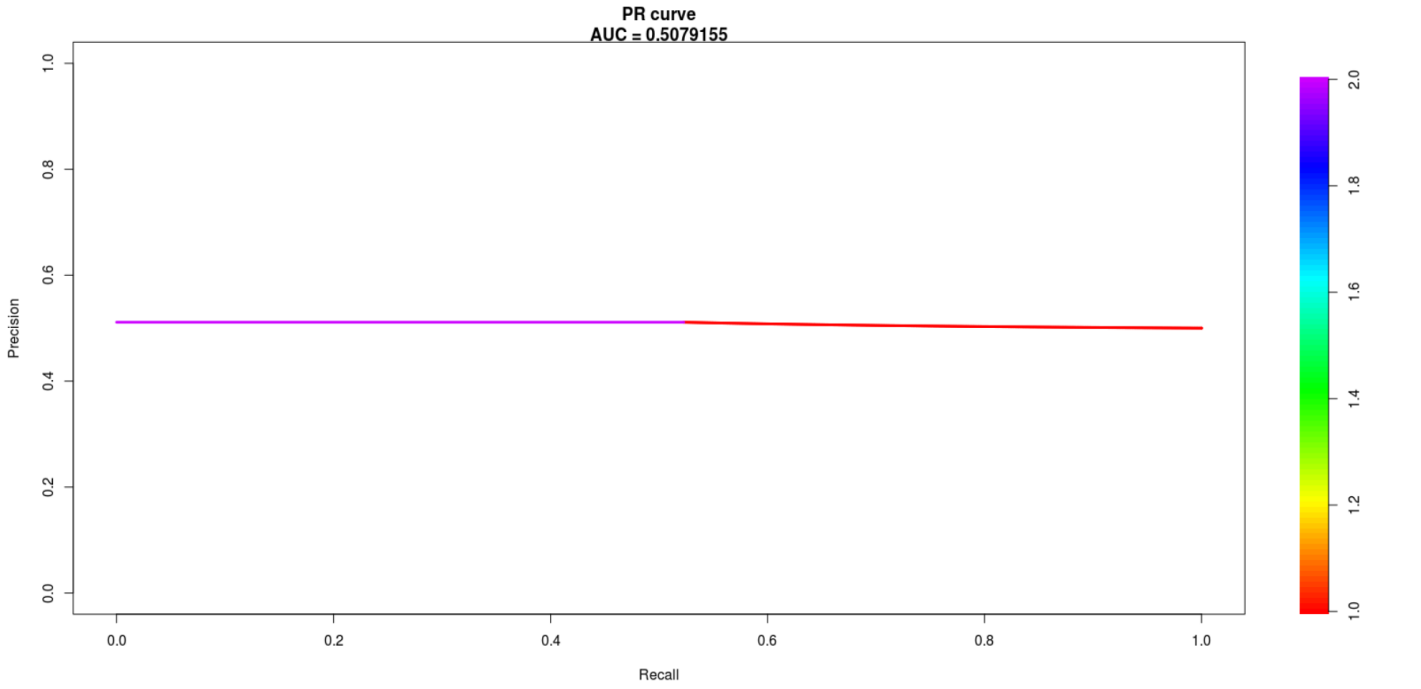
#Combined Sampling

```{r neural net cm,warning=FALSE}

fd_NN_cm <- neuralnet(Class~.,

                      data = fd_balanced_both,

                      linear.output = FALSE,

                      err.fct = 'ce',

                      likelihood = TRUE,hidden = c(2,2))

plot(fd_NN_cm, rep = 'best')

x44=roc.curve(factor(ifelse(predicted.class_c == "0", "0", "1")),

              factor(fd_balanced_both$Class),curve = TRUE)

x44

plot(x44)

#recall
```

```
a44=(pr.curve(factor(ifelse(predicted.class_c == "0", "0", "1")),
              factor(fd_balanced_under$Class),curve = TRUE))
a44
plot(a44)
```




ROC curve
AUC = 0.521088
Sensitivity
FPR
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0


PR curve
AUC = 0.5079155
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0