Homework 4 | Group 5 | SCMT650

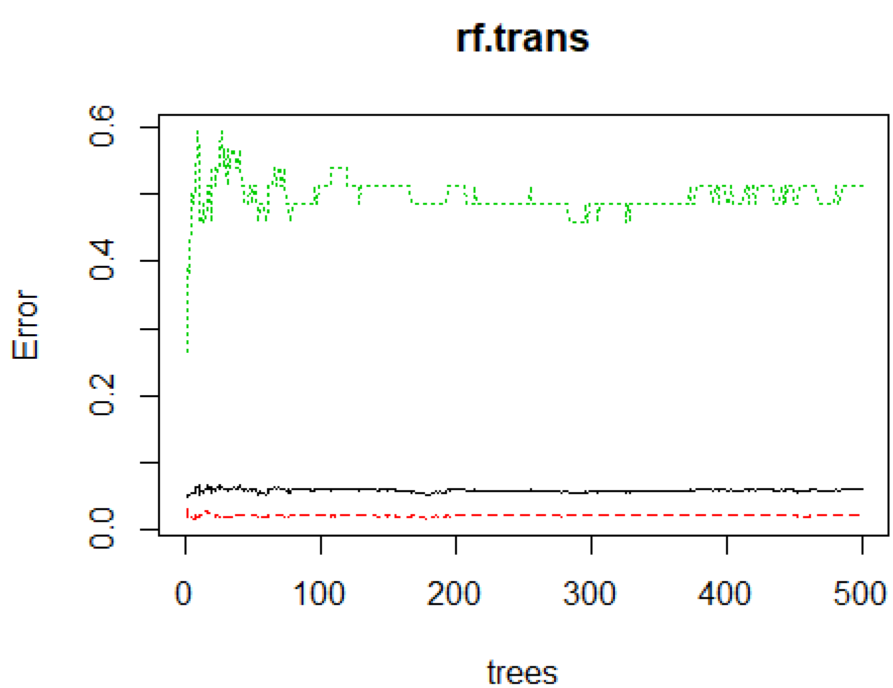
Group 5

4/26/19

```
rm(list=ls())
setwd("E:/R-HW4")

load("transaction.rdata")

library(randomForest)
```

- Question 1: Predicting with Random Forest *
- 1A. classification RF using training data, all variables, 500 trees

```
set.seed(25)
indexes = sample(1:nrow(trans), nrow(trans)/2)
train = trans[indexes,]
test = trans[-indexes,]

rf.trans=randomForest(class~.,data=train,mtry=13,importance=TRUE, ntree=500)
plot(rf.trans)
```




```
rf.trans
```

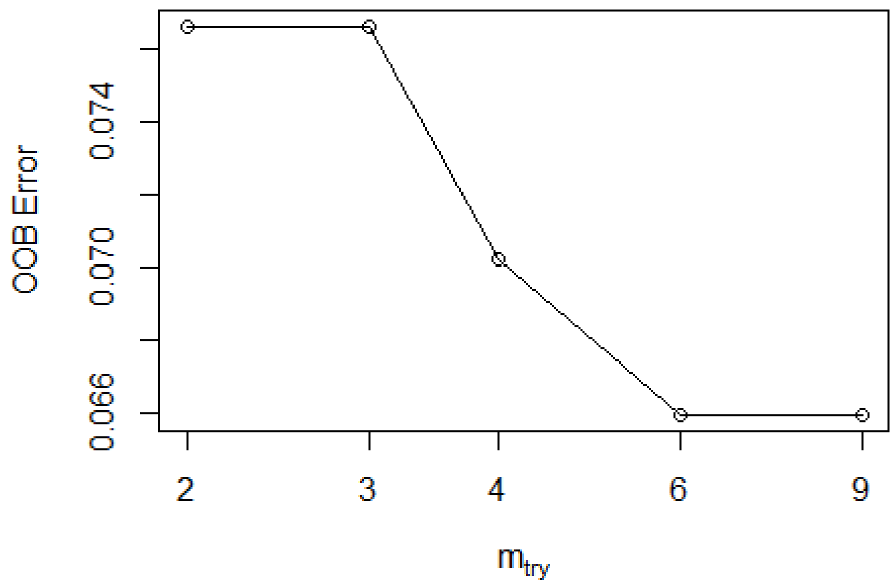
```
##
## Call:
##  randomForest(formula = class ~ ., data = train, mtry = 13, importance =
TRUE,      ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##         OOB estimate of  error rate: 5.96%
## Confusion matrix:
##     0  1 class.error
## 0 424  9  0.02078522
## 1  19 18  0.51351351
```

- 1b. Assessing model accuracy
- Answer: accuracy=0.9468085

```
yhat.trans = predict(rf.trans,newdata=test)
mean(yhat.trans==test$class)

## [1] 0.9468085
```

- 1c. tuning RF on mtry What is the best model?
- Answer: Using tuneRF function the optimal mtry = 6 with an OOB error estimate of 6.38% # How accurate is it? Answer: Accuracy = 0.9404255 (94%)

```
# tuning mtry conirmed using tunrRF function: mtry=6 has lowest OOB error
x <- test[,1:13]
y <- test[,14]
set.seed(25)
bestmtry <- tuneRF(x, y, stepFactor=1.5, improve=1e-5, ntree=500)

## mtry = 3  OOB error = 7.66%
## Searching left ...
## mtry = 2     OOB error = 7.66%
## 0 1e-05
## Searching right ...
## mtry = 4     OOB error = 7.02%
## 0.08333333 1e-05
## mtry = 6     OOB error = 6.6%
## 0.06060606 1e-05
## mtry = 9     OOB error = 6.6%
## 0 1e-05
```

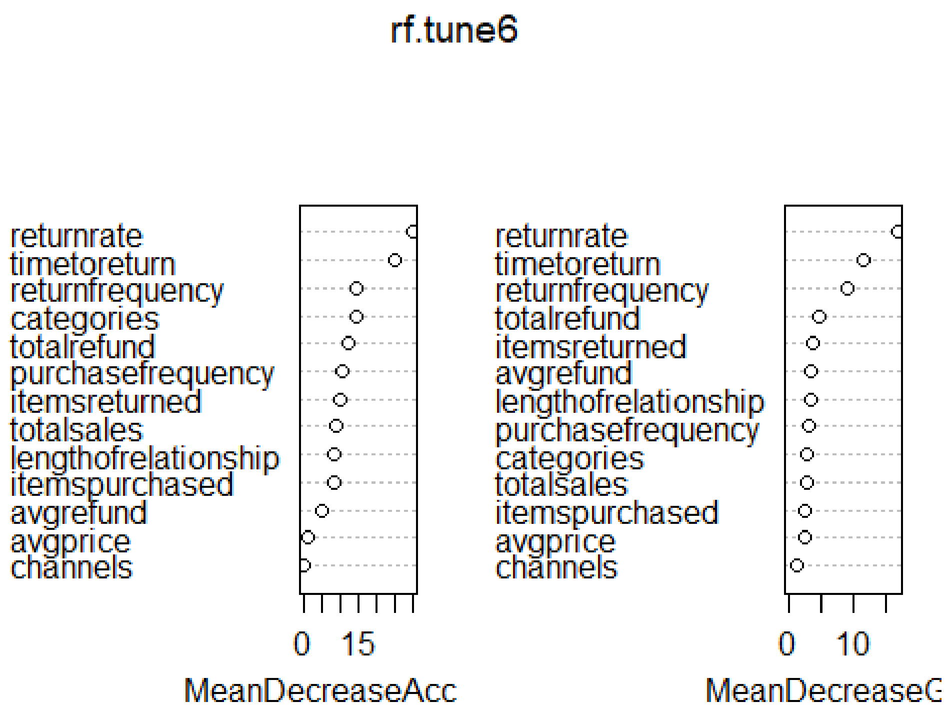



```
print(bestmtry)

##       mtry   OOBError
## 2.OOB    2 0.07659574
## 3.OOB    3 0.07659574
## 4.OOB    4 0.07021277
## 6.OOB    6 0.06595745
## 9.OOB    9 0.06595745

# Accuracy using mtry=6 = 0.9404255
rf.tune6=randomForest(class~.,data=train,mtry=6,importance=TRUE, ntree=500)
rf.tune6

##
## Call:
##  randomForest(formula = class ~ ., data = train, mtry = 6, importance =
TRUE,      ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 6.38%
## Confusion matrix:
##     0  1 class.error
## 0 425  8  0.01847575
## 1  22 15  0.59459459
```

```r
yhat.tune6 = predict(rf.tune6,newdata=test)
mean(yhat.tune6==test$class)

## [1] 0.9404255
```

- 1d. variable importance
- Answer: Using the tuned rf model with mtry=4, the least important variables are avgprice and channels

```r
importance(rf.tune6)

##                                0         1 MeanDecreaseAccuracy
## lengthofrelationship  7.7818548  2.426026            8.1552253
## itemsreturned         9.1135934  2.609188            9.5970201
## itemspurchased        8.2527038 -1.467506            7.9304345
## totalsales            8.9824260 -2.061422            8.8891241
## totalrefund          11.8488262  4.898853           12.3154318
## categories           14.9323467 -2.154041           14.2331088
## timetoreturn         17.7209350 18.947409           25.0615407
## channels              1.5462594 -3.084234            0.0635989
## avgprice              0.9095829  0.704281            1.1991678
## purchasefrequency    10.9939054 -2.139299           10.4605918
## returnfrequency      12.3824400  9.313448           14.4584373
## avgrefund             2.4928142  5.741851            5.0405307
## returnrate           19.3279129 29.832765           30.0434006
##                      MeanDecreaseGini
## lengthofrelationship         3.471108
## itemsreturned                3.688608
## itemspurchased               2.533232
## totalsales                   2.778957
## totalrefund                  4.494319
## categories                   2.780368
## timetoreturn                11.702856
## channels                     1.281861
## avgprice                     2.425241
## purchasefrequency            3.006307
## returnfrequency              9.038900
## avgrefund                    3.499140
## returnrate                  17.005578

varImpPlot(rf.tune6)
```



- 1e. Dropping the least important variables (avgprice, channels)
- Answer: these variables were dropped since they have the lowest importance from the MeanDecreaseAccuracy & MeanDecreaseGini plot # The accuracy increases from 94.04% to 94.89%

```
rf.importance=randomForest(class~.-avgprice-
channels,data=train,mtry=6,importance=TRUE, ntree=500)
rf.importance

##
## Call:
##  randomForest(formula = class ~ . - avgprice - channels, data = train,
mtry = 6, importance = TRUE, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 5.74%
## Confusion matrix:
##     0  1 class.error
## 0 427   6  0.01385681
## 1  21 16  0.56756757

yhat.importance = predict(rf.importance,newdata=test)
mean(yhat.importance==test$class)

## [1] 0.9489362
```

- Question 2: Predicting with Support Vector Machines *

```
rm(list=ls())
("E:/R-HW4")

## [1] "E:/R-HW4"

library(e1071)

load("transaction.rdata")

# Test and Train datasets
set.seed(25)
trainindex=sample(nrow(trans),trunc(nrow(trans)/2))
train=trans[trainindex,]
test=trans[-trainindex,]
```

- 2a. SVM [tuning on the cost parameter for values of 0.1, 1, 5, 10, 50, and 100]
- What is the best tuned value for cost? ANSWER: 0.1
- How many support vectors are associated with this model? ANSWER: 70 support vectors
- What is the test error rate? ANSWER: Accuracy 0.9149, Test Error Rate is 0.0851

```
set.seed(25)
tune.out1=tune(svm,class~.,data=train,kernel="linear",ranges=list(cost=c(0.1,
1,5,10,50,100)))
summary(tune.out1)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.07021277
##
## - Detailed performance results:
##    cost      error dispersion
## 1   0.1 0.07021277 0.03759534
## 2   1.0 0.07872340 0.05216504
## 3   5.0 0.07872340 0.04601738
## 4  10.0 0.07872340 0.04601738
## 5  50.0 0.08085106 0.04463016
## 6 100.0 0.08085106 0.04463016

bestmod1=tune.out1$best.model
summary(bestmod1)
```

```
##
## Call:
## best.tune(method = svm, train.x = class ~ ., data = train, ranges =
list(cost = c(0.1,
##     1, 5, 10, 50, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.07692308
##
## Number of Support Vectors:  70
##
##  ( 37 33 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
predict.y1=predict(bestmod1,test)
table(predict.y1,test$class)
```

```
##
## predict.y1   0   1
##          0 410  35
##          1   5  20
```

```r
mean(predict.y1==test$class)
```

```
## [1] 0.9148936
```

- 2b. SVM radial kernel and tuning with respect to values of gamma=0.5, 1, 2, 3, and 4.
- Answer: The best tuned value for cost is 5 and the best gamma is 0.5.
- There are 256 support vectors associated with this model.
- The accuracy is 0.8894, so the test error rate is 0.1106.

```r
set.seed(25)
tune.out2=tune(svm,class~.,data=train,kernel="radial",ranges=list(cost=c(0.1,
1,5,10,50,100),gamma=c(0.5,1,2,3,4)))
summary(tune.out2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
```

```
##  cost gamma
##     5   0.5
##
## - best performance: 0.07659574
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1    0.1   0.5 0.07872340 0.03481684
## 2    1.0   0.5 0.07872340 0.03481684
## 3    5.0   0.5 0.07659574 0.03042214
## 4   10.0   0.5 0.07659574 0.03042214
## 5   50.0   0.5 0.07659574 0.03042214
## 6  100.0   0.5 0.07659574 0.03042214
## 7    0.1   1.0 0.07872340 0.03481684
## 8    1.0   1.0 0.07872340 0.03481684
## 9    5.0   1.0 0.07872340 0.03481684
## 10  10.0   1.0 0.07872340 0.03481684
## 11  50.0   1.0 0.07872340 0.03481684
## 12 100.0   1.0 0.07872340 0.03481684
## 13   0.1   2.0 0.07872340 0.03481684
## 14   1.0   2.0 0.07872340 0.03481684
## 15   5.0   2.0 0.07872340 0.03481684
## 16  10.0   2.0 0.07872340 0.03481684
## 17  50.0   2.0 0.07872340 0.03481684
## 18 100.0   2.0 0.07872340 0.03481684
## 19   0.1   3.0 0.07872340 0.03481684
## 20   1.0   3.0 0.07872340 0.03481684
## 21   5.0   3.0 0.07872340 0.03481684
## 22  10.0   3.0 0.07872340 0.03481684
## 23  50.0   3.0 0.07872340 0.03481684
## 24 100.0   3.0 0.07872340 0.03481684
## 25   0.1   4.0 0.07872340 0.03481684
## 26   1.0   4.0 0.07872340 0.03481684
## 27   5.0   4.0 0.07872340 0.03481684
## 28  10.0   4.0 0.07872340 0.03481684
## 29  50.0   4.0 0.07872340 0.03481684
## 30 100.0   4.0 0.07872340 0.03481684
```

```r
bestmod2=tune.out2$best.model
summary(bestmod2)
```

```
##
## Call:
## best.tune(method = svm, train.x = class ~ ., data = train, ranges =
list(cost = c(0.1,
##     1, 5, 10, 50, 100), gamma = c(0.5, 1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
```

```
##  SVM-Kernel:  radial
##        cost:  5
##       gamma:  0.5
##
## Number of Support Vectors:  256
##
##  ( 219 37 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
predict.y2=predict(bestmod2,test)
table(predict.y2,test$class)
```

```
##
## predict.y2   0   1
##          0 411  48
##          1   4   7
```

```
mean(predict.y2==test$class)
```

```
## [1] 0.8893617
```

- 2c.SVM polynomial kernel and tuning with respect to values of degree = 2, 3, and 4.
- Answer: The best tuned value for cost is 5 and the best degree is 2.
- There are 78 support vectors associated with this model.
- The accuracy is 0.9213, so the test error rate is 0.0787.

```
set.seed(25)
tune.out3=tune(svm,class~.,data=train,kernel="polynomial",ranges=list(cost=c(
0.1, 1,5,10,50,100),degree=c(2,3,4)))
summary(tune.out3)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##     5      2
##
## - best performance: 0.05957447
##
## - Detailed performance results:
##     cost degree      error dispersion
## 1    0.1      2 0.06808511 0.03725936
## 2    1.0      2 0.07234043 0.03356641
```

```
## 3    5.0      2 0.05957447 0.03296156
## 4   10.0      2 0.06382979 0.04011953
## 5   50.0      2 0.06808511 0.02615474
## 6  100.0      2 0.07446809 0.03364125
## 7    0.1      3 0.07234043 0.03503288
## 8    1.0      3 0.06808511 0.03296156
## 9    5.0      3 0.06808511 0.03296156
## 10  10.0      3 0.07021277 0.02845731
## 11  50.0      3 0.08723404 0.03242306
## 12 100.0      3 0.07872340 0.02663118
## 13   0.1      4 0.07446809 0.04513448
## 14   1.0      4 0.07021277 0.04815389
## 15   5.0      4 0.06808511 0.03296156
## 16  10.0      4 0.07659574 0.03503288
## 17  50.0      4 0.08510638 0.02456810
## 18 100.0      4 0.09574468 0.03050470
```

```
bestmod3=tune.out3$best.model
summary(bestmod3)
```

```
##
## Call:
## best.tune(method = svm, train.x = class ~ ., data = train, ranges =
list(cost = c(0.1,
##     1, 5, 10, 50, 100), degree = c(2, 3, 4)), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  5
##      degree:  2
##       gamma:  0.07692308
##      coef.0:  0
##
## Number of Support Vectors:  78
##
##  ( 48 30 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
predict.y3=predict(bestmod3,test)
table(predict.y3,test$class)
```

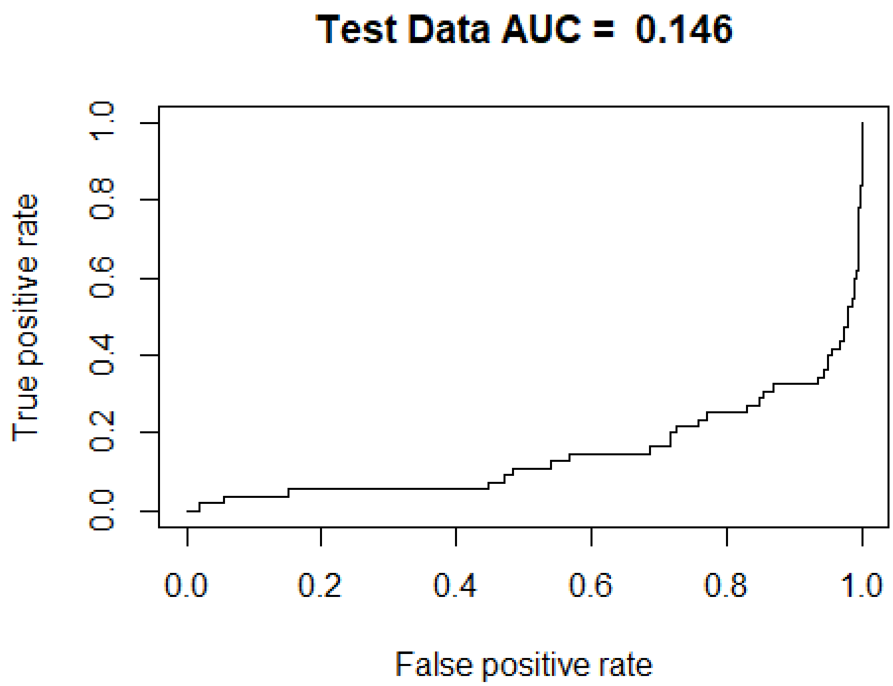
```
##
## predict.y3   0   1
```

```
##          0 411  33
##          1   4  22

mean(predict.y3==test$class)

## [1] 0.9212766
```

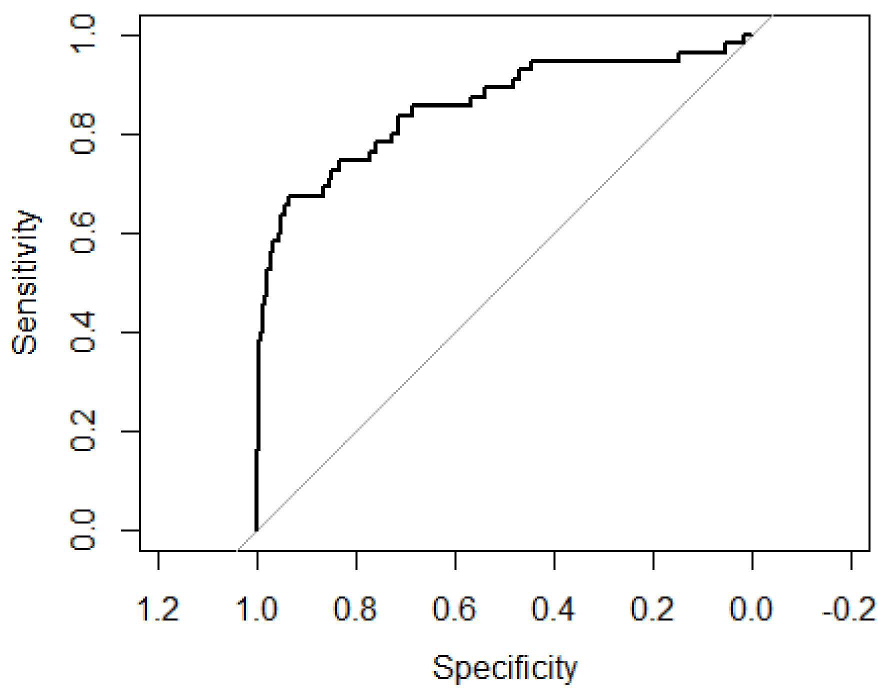
- 2d. what is your best model?
- Answer: Linear kernel - The accuracy is 0.9149, so the test error rate is 0.0851.
- Radial kernel - The accuracy is 0.8894, so the test error rate is 0.1106.
- Polynomial - The accuracy is 0.9213, so the test error rate is 0.0787.
- Since the polynomial model has the highest accuracy and the lowest test error rate, the polynomial model is the best model.

- 2e. Plot an ROC curve for your best model. Is this a good model for predicting return abuse?
- Comment: using the ROCR package, the plot produced is seemingly incorrect [AUC = 0.146], using different axis (FP,TP) than other examples of ROC curves which use (specificity [TN], sensitivity [TP]). However, when using the pROC package, a familiar ROC curve is produced with AUC = 0.8546.
- Answer: The best SVM model with polynomial kernel is 92% accurate at predicting the class value. combined with a AUC value of 0.85 which is approaching 1.

```
library(ROCR)

library(pROC)

tune.out=tune.out3
bestmod=tune.out3$best.model

rocplot=function(pred, truth, ...){
  predob = prediction(pred, truth)
  perf = performance(predob, "tpr", "fpr")
  auc<-performance(predob,measure="auc")@y.values[[1]]
  title<-paste(...,"AUC = ",round(auc, digits=3))
  plot(perf,main=title)}

svmfit.opt=svm(class~., data=trans[trainindex,],
kernel="polynomial",cost=5,degree=2,decision.values=T)
fitted=as.numeric(attributes(predict(svmfit.opt,trans[-
trainindex,],decision.values=T))$decision.values)
rocplot(fitted,trans[-trainindex,"class"],main="Test Data")
```

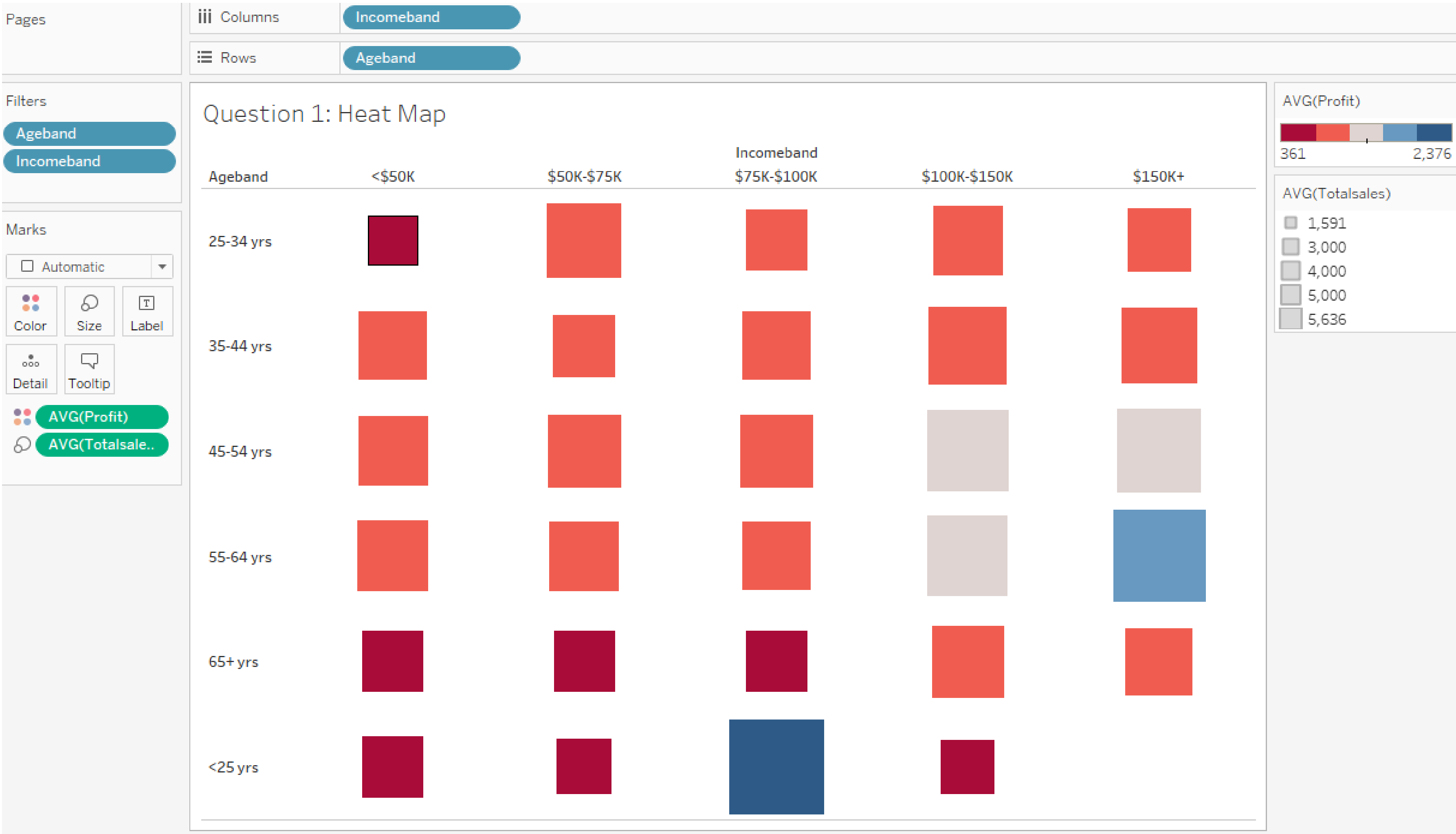



```
#Correct ROC / AUC graph using pROC package
svm.roc = roc(trans[-trainindex,"class"],fitted)
svm.roc

##
## Call:
## roc.default(response = trans[-trainindex, "class"], predictor = fitted)
##
## Data: fitted in 415 controls (trans[-trainindex, "class"] 0) > 55 cases
(trans[-trainindex, "class"] 1).
## Area under the curve: 0.8543

plot(svm.roc)
```

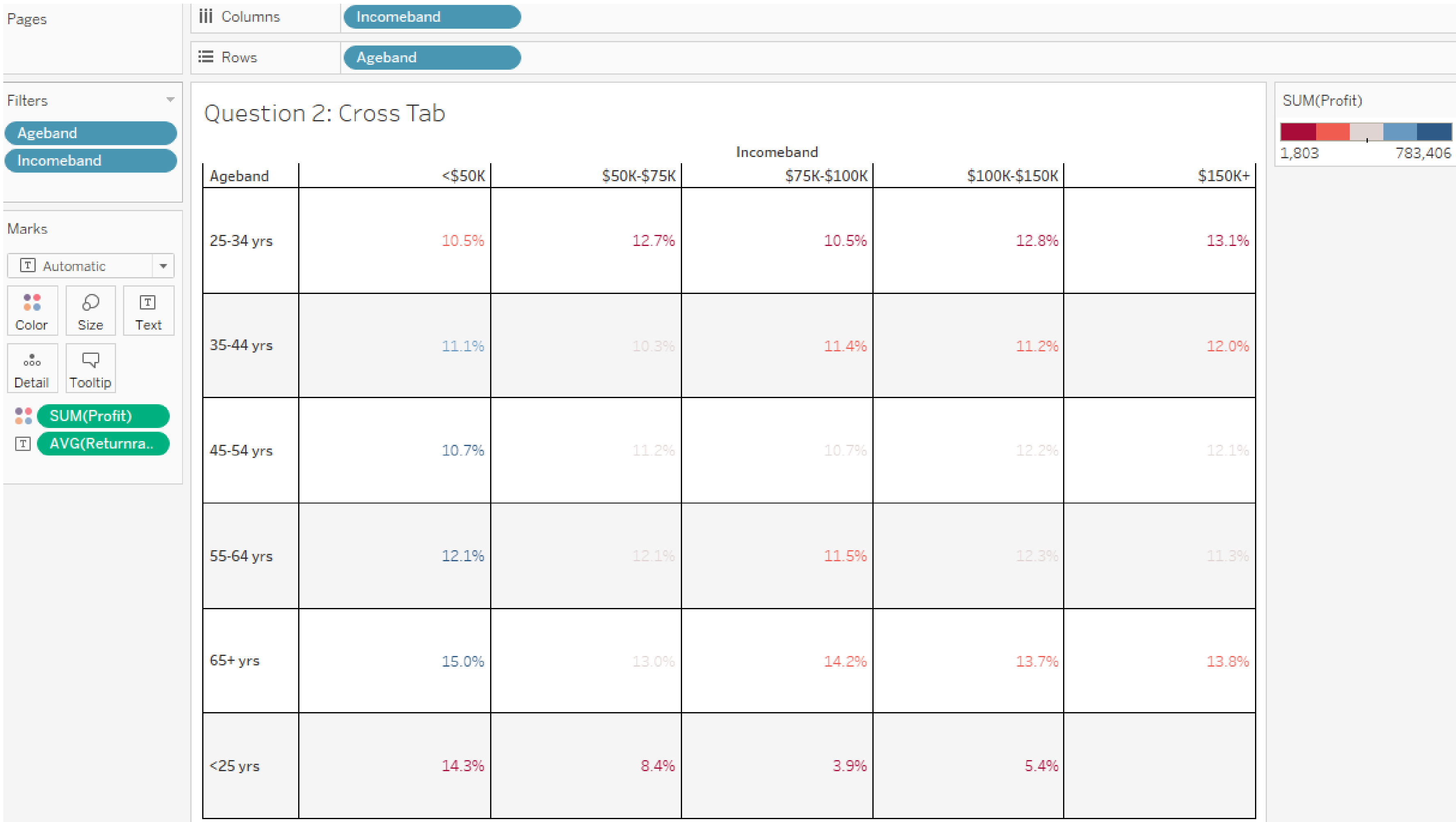

Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
Specificity

Team 5 – Homework 4 - Tableau Section

Q1

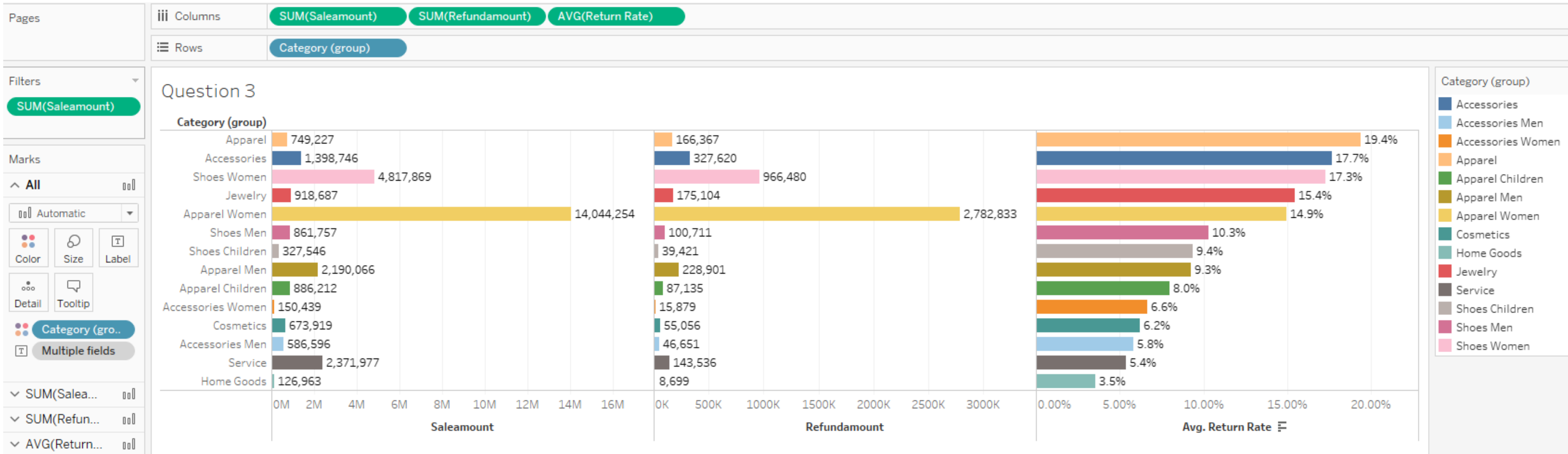


Q2

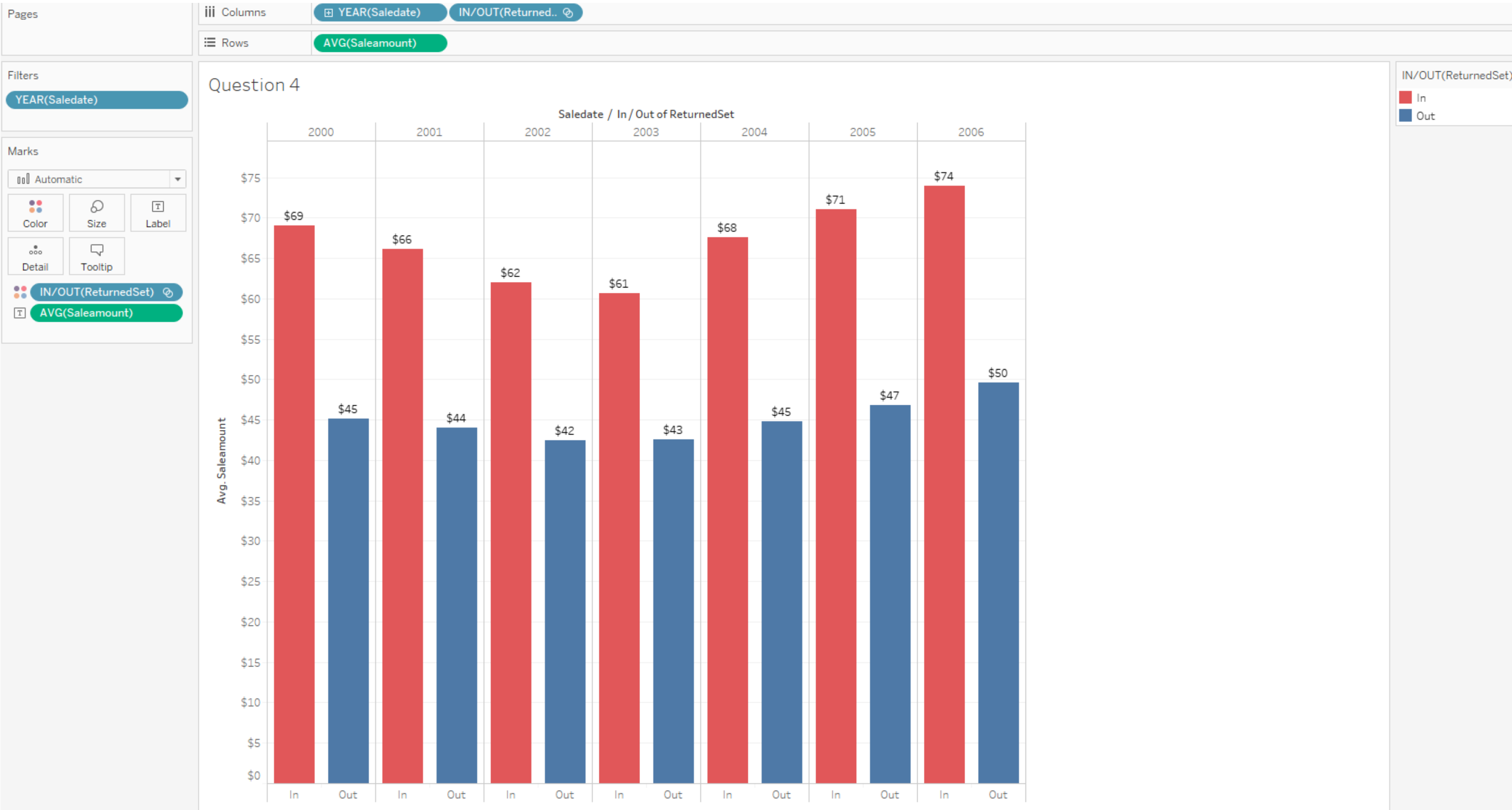


	Incomeband				
Ageband	<$50K	$50K-$75K	$75K-$100K	$100K-$150K	$150K+
25-34 yrs	10.5%	12.7%	10.5%	12.8%	13.1%
35-44 yrs	11.1%	10.3%	11.4%	11.2%	12.0%
45-54 yrs	10.7%	11.2%	10.7%	12.2%	12.1%
55-64 yrs	12.1%	12.1%	11.5%	12.3%	11.3%
65+ yrs	15.0%	13.0%	14.2%	13.7%	13.8%
<25 yrs	14.3%	8.4%	3.9%	5.4%	

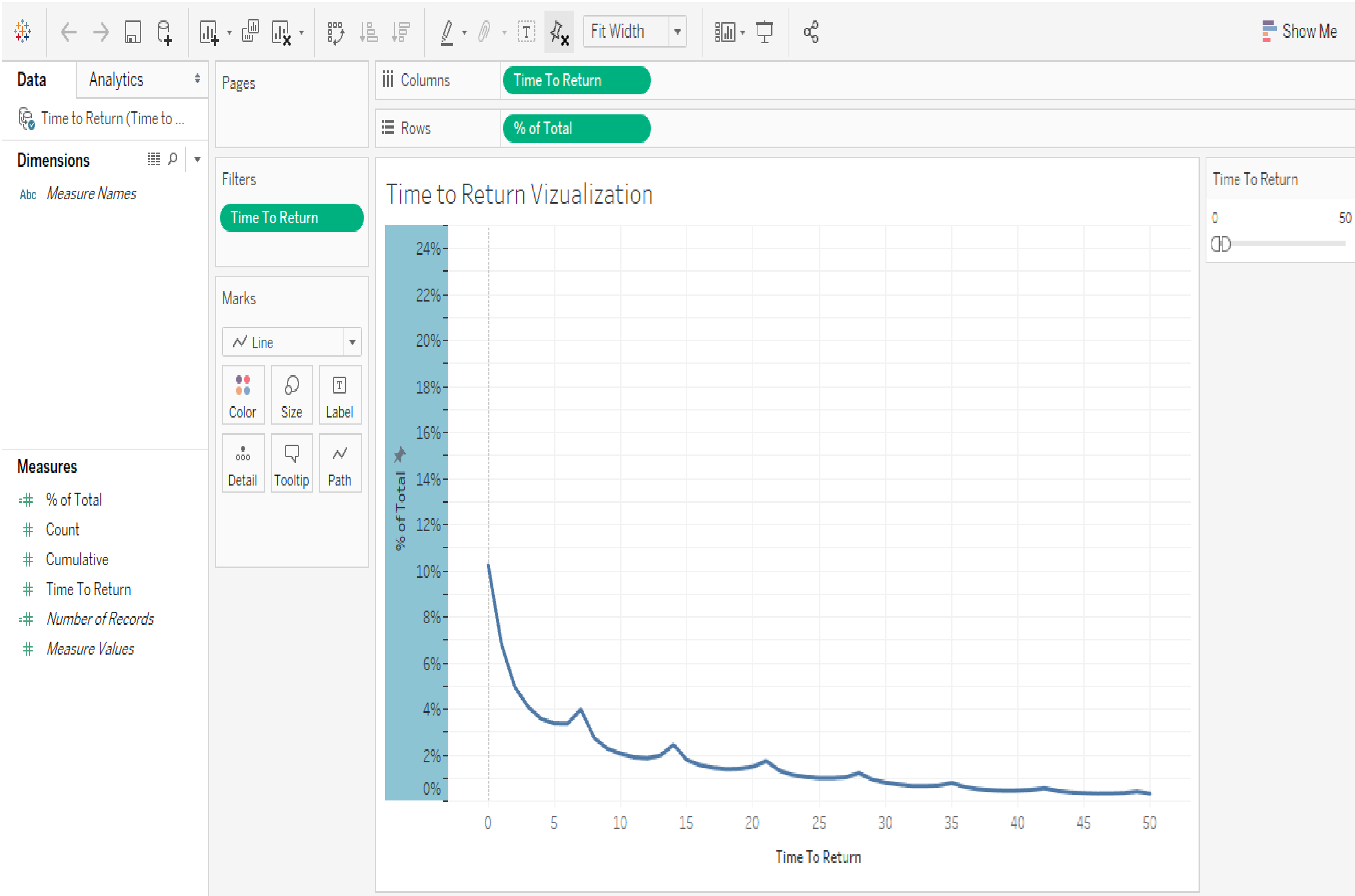
Q3


Pages
Columns
SUM(Saleamount)
SUM(Refundamount)
AVG(Return Rate)
Rows
Category (group)
Filters
SUM(Saleamount)
Marks
All
Automatic
Color
Size
Label
Detail
Tooltip
Category (gro..
Multiple fields
SUM(Salea...
SUM(Refun...
AVG(Return...
Question 3
Category (group)
Apparel 749,227 166,367 19.4%
Accessories 1,398,746 327,620 17.7%
Shoes Women 4,817,869 966,480 17.3%
Jewelry 918,687 175,104 15.4%
Apparel Women 14,044,254 2,782,833 14.9%
Shoes Men 861,757 100,711 10.3%
Shoes Children 327,546 39,421 9.4%
Apparel Men 2,190,066 228,901 9.3%
Apparel Children 886,212 87,135 8.0%
Accessories Women 150,439 15,879 6.6%
Cosmetics 673,919 55,056 6.2%
Accessories Men 586,596 46,651 5.8%
Service 2,371,977 143,536 5.4%
Home Goods 126,963 8,699 3.5%
0M 2M 4M 6M 8M 10M 12M 14M 16M
Saleamount
0K 500K 1000K 1500K 2000K 2500K 3000K
Refundamount
0.00% 5.00% 10.00% 15.00% 20.00%
Avg. Return Rate
Category (group)
Accessories
Accessories Men
Accessories Women
Apparel
Apparel Children
Apparel Men
Apparel Women
Cosmetics
Home Goods
Jewelry
Service
Shoes Children
Shoes Men
Shoes Women

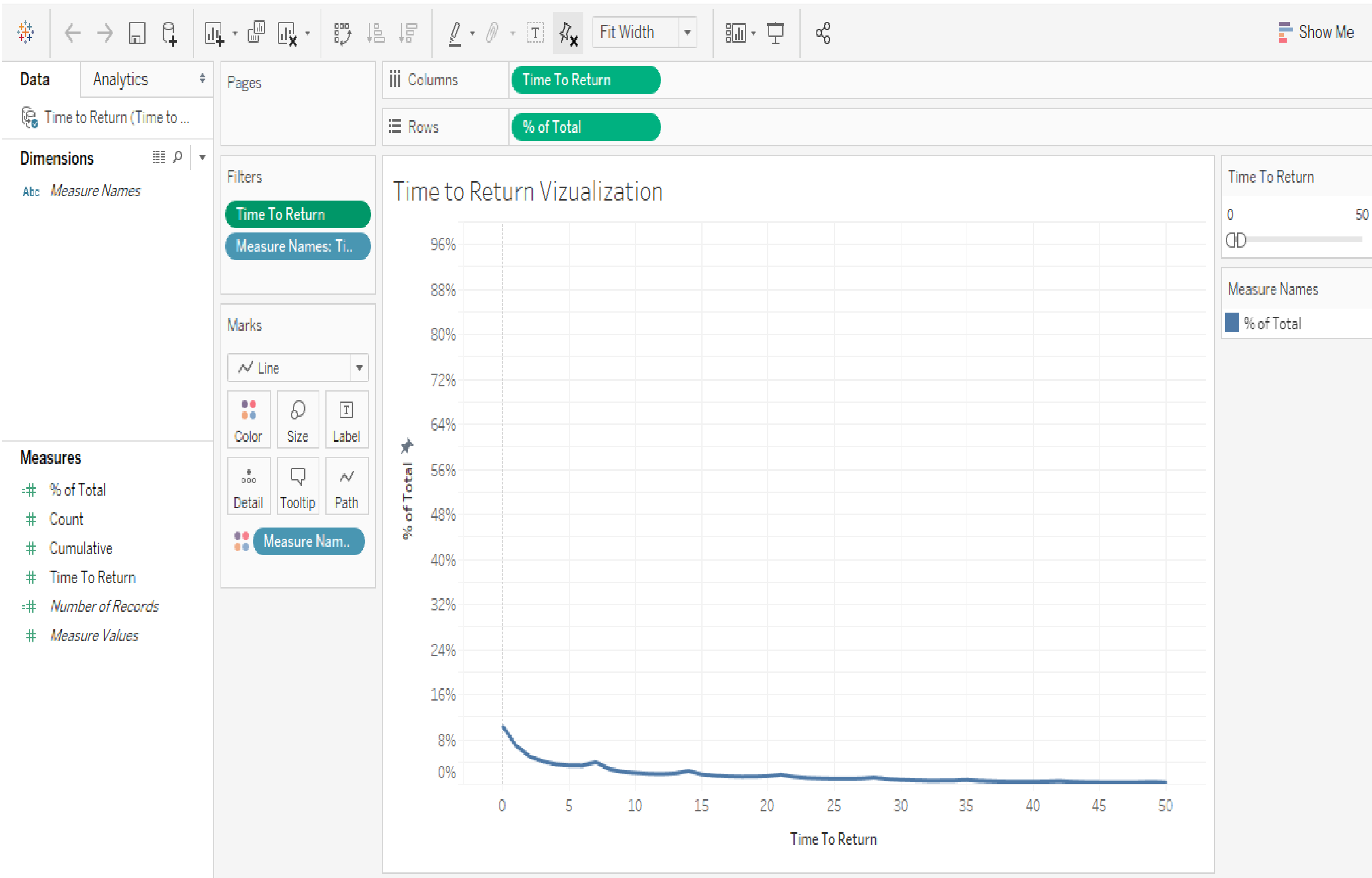
Q4


Pages
Columns
YEAR(Saledate)
IN/OUT(Returned..
Rows
AVG(Saleamount)
Filters
YEAR(Saledate)
Marks
Automatic
Color
Size
Label
Detail
Tooltip
IN/OUT(ReturnedSet)
AVG(Saleamount)
Question 4
Saledate / In / Out of ReturnedSet
2000
2001
2002
2003
2004
2005
2006
Avg. Saleamount
$0
$5
$10
$15
$20
$25
$30
$35
$40
$45
$50
$55
$60
$65
$70
$75
$69
$45
$66
$44
$62
$42
$61
$43
$68
$45
$71
$47
$74
$50
In
Out
IN/OUT(ReturnedSet)
In
Out

Q5


Fit Width
Show Me
Data
Analytics
Time to Return (Time to ...
Dimensions
Measure Names
Measures
% of Total
Count
Cumulative
Time To Return
Number of Records
Measure Values
Pages
Filters
Time To Return
Marks
Line
Color
Size
Label
Detail
Tooltip
Path
Columns
Time To Return
Rows
% of Total
Time to Return Vizualization
% of Total
24%
22%
20%
18%
16%
14%
12%
10%
8%
6%
4%
2%
0%
0
5
10
15
20
25
30
35
40
45
50
Time To Return
Time To Return
0
50

Q6



Show Me
Fit Width
Data
Analytics
Time to Return (Time to ...
Dimensions
Measure Names
Measures
% of Total
Count
Cumulative
Time To Return
Number of Records
Measure Values
Pages
Filters
Time To Return
Measure Names: Ti..
Marks
Line
Color
Size
Label
Detail
Tooltip
Path
Measure Nam..
Columns
Time To Return
Rows
% of Total
Time to Return Vizualization
96%
88%
80%
72%
64%
56%
48%
40%
32%
24%
16%
8%
0%
% of Total
0
5
10
15
20
25
30
35
40
45
50
Time To Return
Time To Return
0
50
Measure Names
% of Total