Module 3 Exercise Questions

Filipp Krasovsky

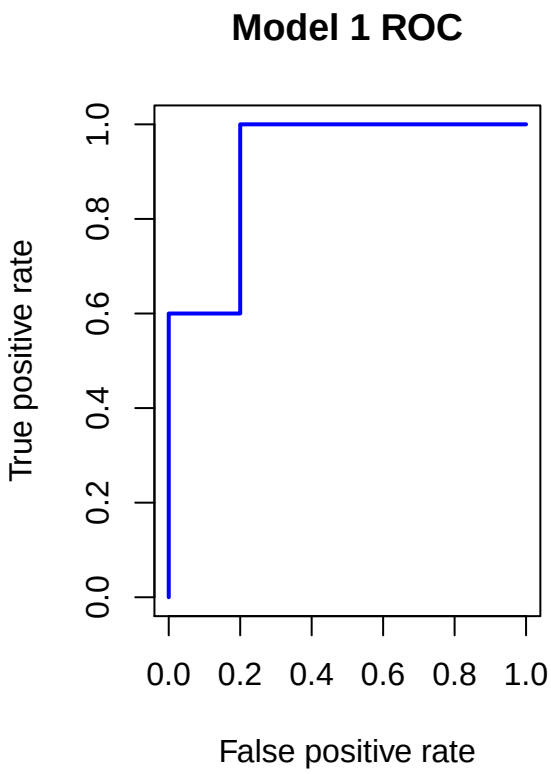
3/20/2021

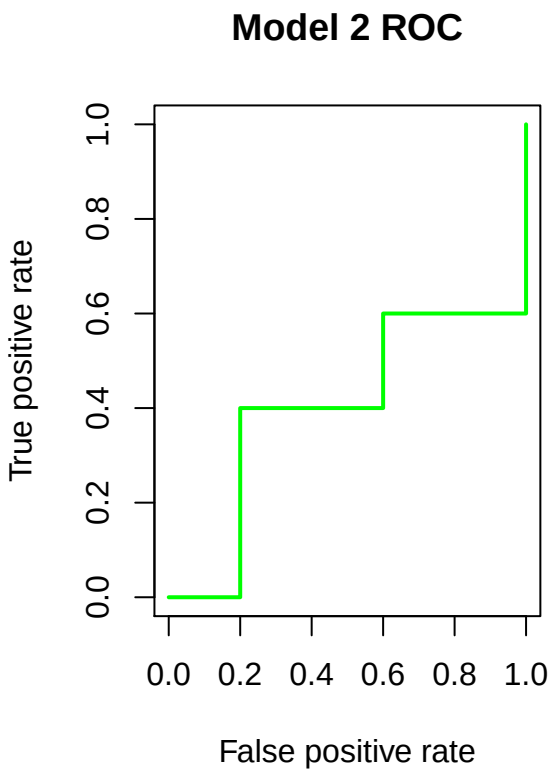
Introduction to Data Mining: Questions 16, 21

16. You are asked to evaluate the performance of two classification models, M1 and M2. The test set you have chosen contains 26 binary attributes, labeled as A through Z. Table 4.13 shows the posterior probabilities obtained by applying the models to the test set.

    a. Plot the ROC curve for both and . (You should plot them on the same graph.) Which model do you think is better? Explain your reasons.

```
#load in our dataset and adjust corrupted column headers
setwd("C:/Users/Filipp/Documents/usd_data_sci/502_data mining/module3")
df = read.csv("question_16_data.csv")
names(df)[1] = "models"
#calculate ROC curves for each model
df.1 <- subset(df,df$models=="m1")
pred <- prediction(df.1$predictions,df.1$labels)
perf.1 <- performance(pred,"tpr","fpr")
#..
df.2 <- subset(df,df$model=="m2")
pred <- prediction(df.2$predictions,df.2$labels)
perf.2 <- performance(pred,"tpr","fpr")
#co-plot in the same frame.
par(mfrow=c(1,2))
plot(perf.1,main="Model 1 ROC",col="blue",lwd=2)
plot(perf.2,main="Model 2 ROC",col="green",lwd=2)
```





It's clear that model 1 outperforms model 2 because the area under the curve is significantly closer to 1 than model 2's curve, which seems to perform about as well as randomized guessing.

b.For model M1, suppose you choose the cutoff threshold to be t=0.5. In other words, any test instances whose posterior probability is greater than t will be classified as a positive example. Compute the precision, recall, and F-measure for the model at this threshold value.

```
#load in our dataset and adjust corrupted column headers
setwd("C:/Users/Filipp/Documents/usd_data_sci/502_data mining/module3")
df = read.csv("question_16_data.csv")
names(df)[1] = "models"
#fetch model 1
df.1 <- subset(df.1,df$model=="m1")

#determine if classed as positive based on t
df.1$positive = df.1$predictions > 0.5

#get TP,FP,FN,TN
TP = nrow(subset(df.1,labels=="+" & positive==TRUE))
FP = nrow(subset(df.1,labels=="-" & positive==TRUE))
FN = nrow(subset(df.1,labels=="+" & positive==FALSE))

#define formulas for precision, recall, and the F-measure
precision <- function(TP,FP){
  return (TP/(TP+FP))
}
```

```
recall <- function(TP,FN){
  return (TP/(TP+FN))
}
f_measure <- function(precision,recall){
  return ( 2 * (precision*recall)/(precision+recall))
}

df.precision = precision(TP,FP)
df.recall    = recall(TP,FN)
df.fmeasure  = f_measure(df.precision,df.recall)

print(df.precision)
```

```
## [1] 0.75
```

```
print(df.recall)
```

```
## [1] 0.6
```

```
print(df.fmeasure)
```

```
## [1] 0.6666667
```

Question 21: Given the data sets shown in Figures 4.59 , explain how the decision tree, naïve Bayes, and k-nearest neighbor classifiers would perform on these data sets.

Dataset 1: KNN would perform poorly given the amount of noise, especially at smaller values of K. Decision tree. Our decision tree would likely do fine in the face of noise and irrelevant or even redundant attributes. Our naive Bayes might have performance degradation - the noise in the dataset is significant enough where it won't be averaged out during the modeling process. Another consideration might be that the distinguishing attributes take on a linear relationship and inhibit performance as well.

Dataset 2: Our decision tree would suffer given that there's less of a clear split boundary for the distinguishing attributes because of all the noise included inside of them. Our KNN classifier would do better for this data due to the fact that there's less noise/irrelevant attributes. Finally, our Bayes classifier would do very poorly due to the attribute dependence.

Dataset 3: Bayes will perform well due to the clear-cut class-conditional dependence for each attribute. KNN will perform without major inhibitors. A decision tree might falter due to the large number of attributes.

Dataset 4: KNN will do well because of the clear boundaries for each class along the x and y attributes. Decision trees will work without any major roadblocks. Bayes will not work because of the attribute dependence.

Dataset 5: KNN will perform well due to the clear boundary split. Decision trees will work but may have a large number of nodes. Bayes may not do well due to attribute dependence.

Dataset 6: KNN will work well, decision trees will need a high node count to navigate the decision boundaries. Bayes may underperform due to attribute dependence.

Data Science Using Python and R: Chapter 5 - Page 78: Questions #28, 29, 30, 31, 32, 33, & 34 We will be using the churn data set for these exercises.

```
#read in our churn dataset
setwd("C:/Users/Filipp/Documents/usd_data_sci/502_data mining/module1/Website Data Sets")
df = read.csv("churn")
#sanity check
head(df)
```

```
##   State Account.Length Area.Code    Phone Intl.Plan VMail.Plan VMail.Message
## 1    KS            128       415 382-4657        no        yes            25
## 2    OH            107       415 371-7191        no        yes            26
## 3    NJ            137       415 358-1921        no         no             0
## 4    OH             84       408 375-9999       yes         no             0
## 5    OK             75       415 330-6626       yes         no             0
## 6    AL            118       510 391-8027       yes         no             0
##   Day.Mins Day.Calls Day.Charge Eve.Mins Eve.Calls Eve.Charge Night.Mins
## 1    265.1       110      45.07    197.4        99      16.78      244.7
## 2    161.6       123      27.47    195.5       103      16.62      254.4
## 3    243.4       114      41.38    121.2       110      10.30      162.6
## 4    299.4        71      50.90     61.9        88       5.26      196.9
## 5    166.7       113      28.34    148.3       122      12.61      186.9
## 6    223.4        98      37.98    220.6       101      18.75      203.9
##   Night.Calls Night.Charge Intl.Mins Intl.Calls Intl.Charge CustServ.Calls
## 1          91        11.01      10.0          3        2.70              1
## 2         103        11.45      13.7          3        3.70              1
## 3         104         7.32      12.2          5        3.29              0
## 4          89         8.86       6.6          7        1.78              2
## 5         121         8.41      10.1          3        2.73              3
## 6         118         9.18       6.3          6        1.70              0
##   Old.Churn Churn
## 1    False. False
## 2    False. False
## 3    False. False
## 4    False. False
## 5    False. False
## 6    False. False
```

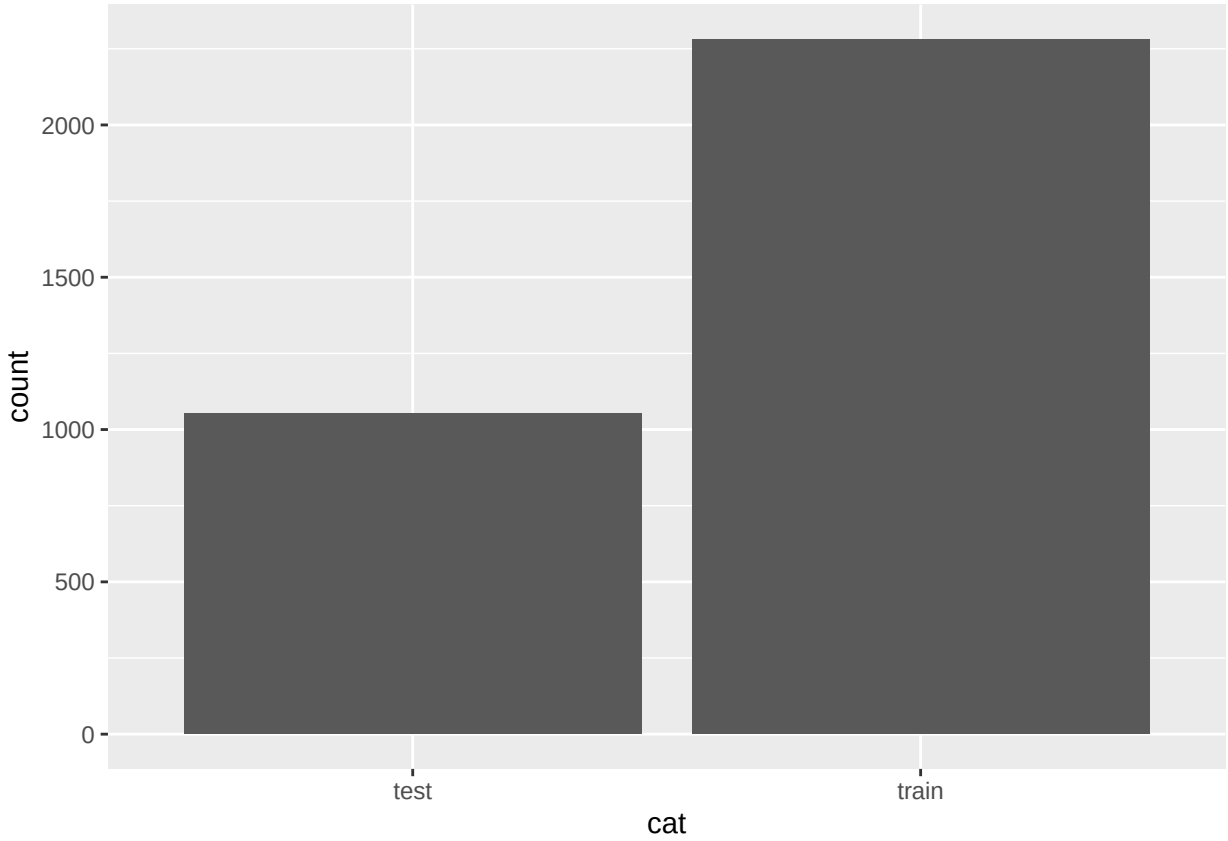
Question 28: Partition the data set, so that 67% of the records are included in the training data set and 33% are included in the test data set. Use a bar graph to confirm your proportions.

```
n <- dim(df)[1]
train_ind <- runif(n) < 0.67
df_train <- df[train_ind,]
df_test  <- df[!train_ind,]

df_train$cat = "train"
df_test$cat  = "test"

df_new <- rbind(df_train, df_test)

library(ggplot2)
ggplot(df_new, aes(cat)) + geom_bar()
```



The bar graphs validate our partition.

Question 29

Identify the total number of records in the training data set and how many records in the training data set have a churn value of true.

We can do this by constructing a contingency table:

```
        #the addmargins function creates a row and column to table A=t.v1 called "total" which contains
        t.v1 <- table(df_train$Churn)
        t.v2 <- addmargins(A = t.v1, FUN = list(total = sum),quiet = TRUE)
        #print with percentages and without
        print(t.v2)
```

```
##
## False  True total
##  1947   334  2281
```

Question 30

Use your answers from the previous exercise to calculate how many true churn records you need to resample in order to have 20% of the rebalanced data set have true churn values.

```
#formula
resample_size <- function(n,p,rare){
  return (((p*n)-rare)/(1-p))
}
```

```
resample_size(nrow(df_train),0.20,338)
```

```
## [1] 147.75
```

We need 148 resampled instances to have a 20% churn=true proportion.

Question 31

Perform the rebalancing described in the previous exercise and confirm that 20% of the records in the rebalanced data set have true churn values.

```
to.resample<- which(df_train$Churn =="True") #get row numbers that correspond to cond
our.resample<- sample(x=to.resample, size=148, replace=TRUE)

our.resample <- df_train[our.resample,]
df_train_rebal <- rbind(df_train,our.resample)

#confirm:
t.v1 <- table(df_train_rebal$Churn)
t.v2 <- rbind(t.v1, round(prop.table(t.v1), 4))
colnames(t.v2) <- c("Churn = False", "Churn = True");
rownames(t.v2) <- c("Count", "Proportion")
t.v2
```

```
##            Churn = False Churn = True
## Count           1947.0000     482.0000
## Proportion         0.8016       0.1984
```

32. Which baseline model do we use to compare our classification model performance against? To which value does this baseline model assign all predictions? What is the accuracy of this baseline model?

We use the all negative (ie Churn = false) model as our baseline. This model assigns all predictions to the value "no" for churn. Our accuracy depends on whether or not our dataset is rebalanced - if it is, then we have 80% accuracy. Otherwise, our baseline model is 85% accurate.

33. Validate your partition by testing for the difference in mean day minutes for the training set versus the test set.

```
t.test(x = df_train$Day.Mins,y = df_test$Day.Mins)
```

```
##
##  Welch Two Sample t-test
##
## data:  df_train$Day.Mins and df_test$Day.Mins
## t = 0.32428, df = 2023.5, p-value = 0.7458
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.336622  4.658655
## sample estimates:
## mean of x mean of y
##  179.9837  179.3227
```

At an alpha of 0.05, we do not have enough evidence to reject the null hypothesis that the two datasets are systematically different on the axis of day minutes.

34. Validate your partition by testing for the difference in proportion of true churn records for the training set versus the test set.

```
train_x = nrow(subset(df_train,Churn=="True"))
test_x  = nrow(subset(df_test,Churn =="True"))
prop.test(x=c(train_x,test_x), n=c(nrow(df_train), nrow(df_test)),conf.level=0.95)
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  c(train_x, test_x) out of c(nrow(df_train), nrow(df_test))
## X-squared = 0.097551, df = 1, p-value = 0.7548
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.02148428  0.03106833
## sample estimates:
##   prop 1   prop 2
## 0.146427 0.141635
```

We do not have enough evidence to reject the null hypothesis that our datasets are systematically different.

Chapter 7 - Page 109: Questions #23, 24, 25, 26, 27, 28, 29, & 30 For the following exercises, work with the adult_ch6_training and adult_ch6_test data sets.

```
setwd("C:/Users/Filipp/Documents/usd_data_sci/502_data mining/module1/Website Data Sets")

train = read.csv("adult_ch6_training")
test =  read.csv("adult_ch6_test")

#factorize
train$Marital.status = factor(train$Marital.status)
#test$Marital.status = factor(test$Marital.status)
train$Income = factor(train$Income)
#test$Income = factor(test$Income)
head(train)
```

```
##   Marital.status Income Cap_Gains_Losses
## 1  Never-married  <=50K          0.02174
## 2       Divorced  <=50K          0.00000
## 3        Married  <=50K          0.00000
## 4        Married  <=50K          0.00000
## 5        Married  <=50K          0.00000
## 6        Married   >50K          0.00000
```

Question 23 Using the training data set, create a C5.0 model (Model 1) to predict a customer's Income using Marital Status and Capital Gains and Losses. Obtain the predicted responses.

```
library(C50)
```

```
## Warning: package 'C50' was built under R version 4.0.4
```

```
#run our model and then point it at our test data
C5 <- C5.0(formula = Income ~ Marital.status + Cap_Gains_Losses, data = train, control = C5.0Control(mi
predict <- (predict.C5.0(C5,newdata=test))
```

Question 24 Evaluate Model 1 using the test data set. Construct a contingency table to compare the actual and predicted values of Income.

```
#since we already did that step in #23, we can move ahead with our table:
results <- data.frame(predict,test$Income)
names(results) = c("predicted","actual")
table(results$predicted,results$actual)
```

```
##
##         <=50K >50K
##   <=50K  4632 1105
##   >50K     42  376
```

Interpretation: of the observed values of Income <=50k, 4632 were correctly classified. Of the observed values of Income >50k, 376 were correctly classified.

Question 25 For Model 1, recapitulate Table 7.4 from the text, calculating all of the model evaluation measures shown in the table. Call this table the Model Evaluation Table. Leave space for Model 2.

For ease, we will target income >50k as 1, and <=50k as zero.

```
results$predicted = ifelse(results$predicted==">50K",1,0)
results$actual    = ifelse(results$actual==">50K",1,0)

tp = length(which(results$predicted==1 & results$actual==1))
tn = length(which(results$predicted==0 & results$actual==0))
fp = length(which(results$predicted==1 & results$actual==0))
fn = length(which(results$predicted==0 & results$actual==1))

accuracy <- function(tp,tn,fp,fn){
  n = sum(tp,tn,fp,fn)
  return(sum(tp,tn)/n)
}
sensitivity <- function(TP,FN){
    return (TP/(TP+FN))
}

specificity <- function(FP,TN){
    return (TN/(FP+TN))
}

precision <- function(TP,FP){
    return (TP/(TP+FP))
}

recall <- function(TN,FP){
    return (TN/(TN+FP))
```

```
}

error_rate <- function(accuracy){
  return (1-accuracy)
}

f_1 <- function(precision,recall){
  numerator = precision * recall
  denominator = precision + recall
  return ( 2 * numerator/denominator)
}

f_2 <- function(precision,recall){
  numerator = precision * recall
  denominator = precision + recall
  return ( 5* (numerator/denominator))
}

f_0.5 <- function(precision,recall){
  numerator = precision * recall
  denominator = precision + recall
  return ( 1.25 * numerator/denominator)
}

a = accuracy(tp,tn,fp,fn)
se=sensitivity(tp,fn)
sp=specificity(fp,tn)
p = precision(tp,fp)
r = recall(tn,fp)
er=error_rate(a)
fone = f_1(p,r)
ftwo = f_2(p,r)
fhalf= f_0.5(p,r)

paste("accuracy",a)
```

```
## [1] "accuracy 0.813647441104793"
```

```
paste("sensitivity",se)
```

```
## [1] "sensitivity 0.25388251181634"
```

```
paste("specificity",sp)
```

```
## [1] "specificity 0.991014120667522"
```

```
paste("precision",p)
```

```
## [1] "precision 0.899521531100478"
```

```
paste("recall",r)
```

```
## [1] "recall 0.991014120667522"
```

```
paste("err. rate",er)
```

```
## [1] "err. rate 0.186352558895207"
```

```
paste("F1",fone)
```

```
## [1] "F1 0.943053931124107"
```

```
paste("F2",ftwo)
```

```
## [1] "F2 2.35763482781027"
```

```
paste("F0.5",fhalf)
```

```
## [1] "F0.5 0.589408706952567"
```

Question 26: Clearly and completely interpret each of the Model 1 evaluation measures from the Model Evaluation Table.

Accuracy: Ability to correctly identify classes as a fraction of the total number of classifications. Our model was 81% accurate.

Sensitivity: Ability to identify records as positive. Our sensitivity is low, so we do not do a good job of identifying "positive instances", or cases of high income individuals.

Specificity: Ability to identify records as negative. Our specificity was high, so we can identify low income individuals well.

Precision: the proportion of true positives to false positives, answers which selected items are relevant. our precision was fairly high, which means most identified high income individuals were correctly picked.

error rate: how inaccurate our model was at identifying either income class correctly. our error rate is fairly high.

F1,2,0.5: haromic weights of precision and recall. F1 weighs these equally, F2 weights recall more, and F0.5 weighs precision more.

Our model does relatively well when we weigh precision and recall equally, incredibly well when we weigh recall more than precision (ie, when we care more about low income identification), and very poorly when we give more weight to high income identification.

Question 27: Create a cost matrix, called the 3x cost matrix, that specifies a false positive is four times as bad as a false negative.

```
#y-real, x-predicted
cost.C5 <- matrix(c(0,4,1,0), byrow = TRUE, ncol=2)
dimnames(cost.C5) <- list(c("0", "1"), c("0", "1"))
cost.C5
```

```
##   0 1
## 0 0 4
## 1 1 0
```

Question 28. Using the training data set, build a C5.0 model (Model 2) to predict a customer's Income using Marital Status and Capital Gains and Losses, using the 3x cost matrix.

Foreword: we will need to manipulate income into a binary variable for ease of calculation. let 1 = high income.

```
#apparently C5 crashes when nonstandard chars are involved, so here I go renaming everything

setwd("C:/Users/Filipp/Documents/usd_data_sci/502_data mining/module1/Website Data Sets")

train = read.csv("adult_ch6_training")
test =  read.csv("adult_ch6_test")

#factorize and redefine as 1,0 to fit our matrix
train$Marital.status = factor(train$Marital.status)
train$Income = factor(train$Income)
train$Income = ifelse(train$Income==">50K",1,0)

#factorize our test response variable to make it comparable during metric calculation
test$Income = factor(test$Income)
test$Income = ifelse(test$Income==">50K",1,0)

names(train) = c("MaritalStatus","Income","CapGainsLosses")
names(test) = c("MaritalStatus","Income","CapGainsLosses")
train$Income = as.factor(train$Income)
train$MaritalStatus = as.factor(train$MaritalStatus)
C5.costs <- C5.0(Income ~ MaritalStatus + CapGainsLosses, data = train,cost=cost.C5,control = C5.0Contr

costs.pred <- predict(object = C5.costs, newdata = test)
results <- data.frame(costs.pred,test$Income)
names(results) <- c("predicted","actual")
```

Question 29. Evaluate your predictions from Model 2 using the actual response values from the test data set. Add Overall Model Cost and Profit per Customer to the Model Evaluation Table. Calculate all the measures from the Model Evaluation Table.

```
#we already calculated the formulae for our performance metrics, so we can just recalculated tp,fp,tn,e
tp = length(which(results$predicted==1 & results$actual==1))
tn = length(which(results$predicted==0 & results$actual==0))
fp = length(which(results$predicted==1 & results$actual==0))
fn = length(which(results$predicted==0 & results$actual==1))

a = accuracy(tp,tn,fp,fn)
se=sensitivity(tp,fn)
sp=specificity(fp,tn)
p = precision(tp,fp)
r = recall(tn,fp)
er=error_rate(a)
fone = f_1(p,r)
ftwo = f_2(p,r)
```

```
fhalf= f_0.5(p,r)

paste("accuracy",a)
```

```
## [1] "accuracy 0.725426482534525"
```

```
paste("sensitivity",se)
```

```
## [1] "sensitivity 0.909520594193113"
```

```
paste("specificity",sp)
```

```
## [1] "specificity 0.667094565682499"
```

```
paste("precision",p)
```

```
## [1] "precision 0.464002755769893"
```

```
paste("recall",r)
```

```
## [1] "recall 0.667094565682499"
```

```
paste("err. rate",er)
```

```
## [1] "err. rate 0.274573517465475"
```

```
paste("F1",fone)
```

```
## [1] "F1 0.54731579849971"
```

```
paste("F2",ftwo)
```

```
## [1] "F2 1.36828949624927"
```

```
paste("F0.5",fhalf)
```

```
## [1] "F0.5 0.342072374062319"
```

```
#overall model cost
# TN*Cost(tn) + FP*Cost(FP) + FN*Cost(FN) + TP*Cost(TP)
overall_cost = (tn*0)+(fp*4)+(fn*1)+(tp*0)

#overall profit per customer
#-(overall_cost)/total
profit_per_customer = -1 * overall_cost / nrow(test)

paste("overall_cost",overall_cost)
```

```
## [1] "overall_cost 6358"
```

```
paste("profit per customer",profit_per_customer)
```

```
## [1] "profit per customer -1.03298131600325"
```

Question 30. Compare the evaluation measures from Model 1 and Model 2 using the 3x cost matrix. Discuss the strengths and weaknesses of each model.

M1 outperforms on accuracy, specificity, precision, error rate, and all three harmonic means. M2 outperforms on sensitivity. In this instance, the ability to avoid false positives is paramount due to our error costs (ie income>50k). Therefore, precision is key. M1 is the better model.

Chapter 8 - Page 126: Questions #31, 32, 33, & 34 For the following exercises, work with the framingham_nb_training and framingham_nb_test data sets. Use either Python or R to solve each problem.

```
#extract, sanity check
setwd("C:/Users/Filipp/Documents/usd_data_sci/502_data mining/module1/Website Data Sets")

train = read.csv("framingham_nb_training.csv")
test =  read.csv("framingham_nb_test.csv")

head(train)
```

```
##   Sex Educ Death
## 1   2    3     0
## 2   2    2     0
## 3   1    1     0
## 4   2    1     0
## 5   2    1     0
## 6   2    3     0
```

```
head(test)
```

```
##   Sex Educ Death
## 1   1    1     0
## 2   1    2     0
## 3   1    3     0
## 4   1    1     0
## 5   2    2     0
## 6   1    1     0
```

```
library(gridExtra)
library(e1071)
```

31. Run the Naïve Bayes classifier to classify persons as living or dead based on sex and education.

```
nb01 <- naiveBayes(formula = Death ~ Sex + Educ,data=train)
nb01
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
```

```
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##    0    1
## 0.45 0.55
##
## Conditional probabilities:
##    Sex
## Y       [,1]      [,2]
##   0 1.591111 0.4921759
##   1 1.440000 0.4968388
##
##    Educ
## Y       [,1]      [,2]
##   0 2.011111 0.9954735
##   1 1.798182 0.9886406
```

32.

Evaluate the Naïve Bayes model on the framingham_nb_test data set. Display the results in a contingency table. Edit the row and column names of the table to make the table more readable. Include a total row and column.

```
ypred <- predict(object = nb01, newdata =test)
results <- data.frame(ypred,test$Death)
names(results)<-c("predicted","actual")

        t.v1 <- table(results$predicted, results$actual,dnn = )
        t.v2 <- addmargins(A = t.v1, FUN = list(total = sum),quiet = TRUE)
        print(t.v2)
```

```
##
##            0    1 total
##   0      203  105   308
##   1      322  370   692
##   total  525  475  1000
```

33.According to your table in the previous exercise, find the following values for the Naïve Bayes model: a. Accuracy b. Error rate

```
tp = length(which(results$predicted==1 & results$actual==1))
tn = length(which(results$predicted==0 & results$actual==0))
fp = length(which(results$predicted==1 & results$actual==0))
fn = length(which(results$predicted==0 & results$actual==1))

a=accuracy(tp,tn,fp,fn)
a
```

```
## [1] 0.573
```

```
(1-a)
```

```
## [1] 0.427
```

34.According to your contingency table, find the following values for the Naïve Bayes model: a. How often it correctly classifies dead persons. b. How often it correctly classifies living persons.

```
   #a.
  tp/nrow(results)
```

```
## [1] 0.37
```

```
  #b.
  tn/nrow(results)
```

```
## [1] 0.203
```