Gabi Rivera

12Nov2022

ADS502-01

**Assignment 3.1: Module 3 Exercise Questions**

Introduction to Data Mining: Exercises 4.14

16. You are asked to evaluate the performance of two classification models, M1and 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. (Only the posterior probabilities for the positive class are shown). As this is a two-class problem, P(−)=1−P(+) and P(−|A, …, Z)=1−P(+|A, …, Z). Assume that we are mostly interested in detecting instances from the positive class.

**Table 4.13. Posterior probabilities for Exercise 16.**

| Instance | True Class | P(+\|A, …, Z, M1) | P(+\|A, …, Z, M2) |
|---|---|---|---|
| 1 | + | 0.73 | 0.61 |
| 2 | + | 0.69 | 0.03 |
| 3 | − | 0.44 | 0.68 |
| 4 | − | 0.55 | 0.31 |
| 5 | + | 0.67 | 0.45 |
| 6 | + | 0.47 | 0.09 |
| 7 | − | 0.08 | 0.38 |
| 8 | − | 0.15 | 0.05 |
| 9 | + | 0.45 | 0.01 |
| 10 | − | 0.35 | 0.04 |

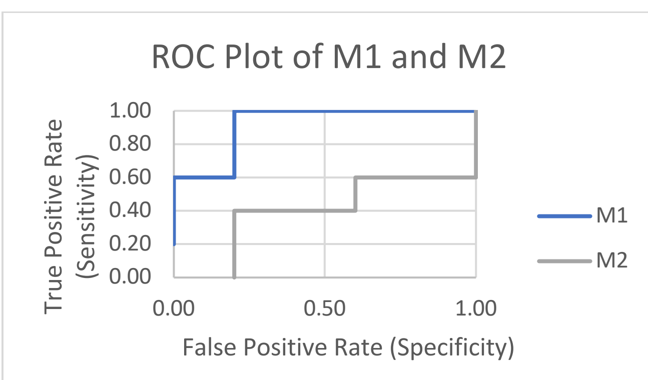
a. Plot the ROC curve for both M1 and M2. (You should plot them on the same graph.)

Which model do you think is better? Explain your reasons.

| True Class | M1 Probabilities (+ class) | TP | FP | TPR = TP/P | FPR = FP/N |
|---|---|---|---|---|---|
| + | 0.73 | 1 | 0 | 0.20 | 0.00 |
| + | 0.69 | 2 | 0 | 0.40 | 0.00 |
| + | 0.67 | 3 | 0 | 0.60 | 0.00 |
| - | 0.55 | 3 | 1 | 0.60 | 0.20 |
| + | 0.47 | 4 | 1 | 0.80 | 0.20 |
| + | 0.45 | 5 | 1 | 1.00 | 0.20 |
| - | 0.44 | 5 | 2 | 1.00 | 0.40 |
| - | 0.35 | 5 | 3 | 1.00 | 0.60 |
| - | 0.15 | 5 | 4 | 1.00 | 0.80 |
| - | 0.08 | 5 | 5 | 1.00 | 1.00 |

| True Class | M2 Probabilities (+ class) | TP | FP | TPR = TP/P | FPR = FP/N |
|---|---|---|---|---|---|
| - | 0.68 | 0 | 1 | 0.00 | 0.20 |
| + | 0.61 | 1 | 1 | 0.20 | 0.20 |
| + | 0.45 | 2 | 1 | 0.40 | 0.20 |
| - | 0.38 | 2 | 2 | 0.40 | 0.40 |
| - | 0.31 | 2 | 3 | 0.40 | 0.60 |
| + | 0.09 | 3 | 3 | 0.60 | 0.60 |
| - | 0.05 | 3 | 4 | 0.60 | 0.80 |
| - | 0.04 | 3 | 5 | 0.60 | 1.00 |
| + | 0.03 | 4 | 5 | 0.80 | 1.00 |
| + | 0.01 | 5 | 5 | 1.00 | 1.00 |

**0.5 decision threshold*



Answer: M1 is the better model because visually it's area under the curve is larger compared to M2 from looking at the Receiver Operating Characteristic plot.

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.

Answer: Precision is 75%, Recall is 60%, and F-measure is 67%.

M1 Confusion Matrix, t=0.5 cutoff decision threshold

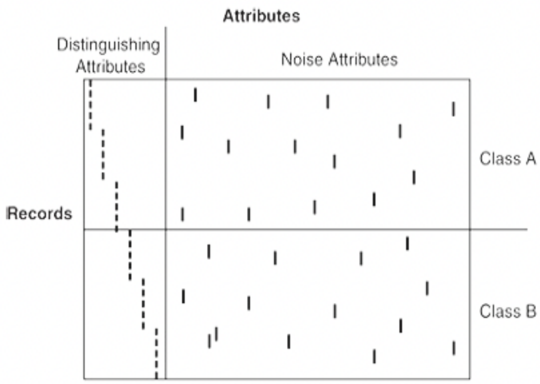
<table>
<tr><td></td><td></td><td colspan="2">Predicted</td></tr>
<tr><td></td><td></td><td>Positive</td><td>Negative</td></tr>
<tr><td rowspan="2">Actual</td><td>Positive</td><td>3</td><td>2</td></tr>
<tr><td>Negative</td><td>1</td><td>4</td></tr>
</table>

Precision = True Positive/(False Positive + True Positive) = ¾ = 75%

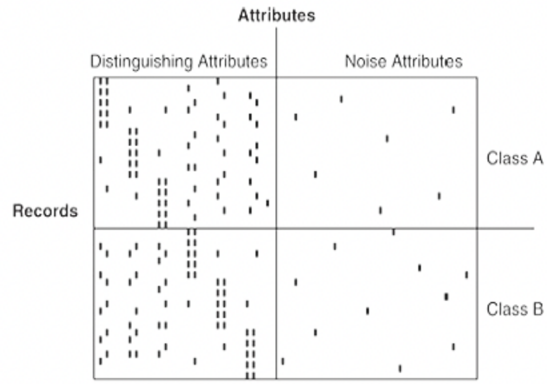
Recall = True Positive Rate (sensitivity) = TP/(TP+FN) = 3/5 = 60%

F-measure = (2*Precision*Recall)/(Recall + Precision) = (2*0.75*0.60)/(0.60*0.75) = 0.67

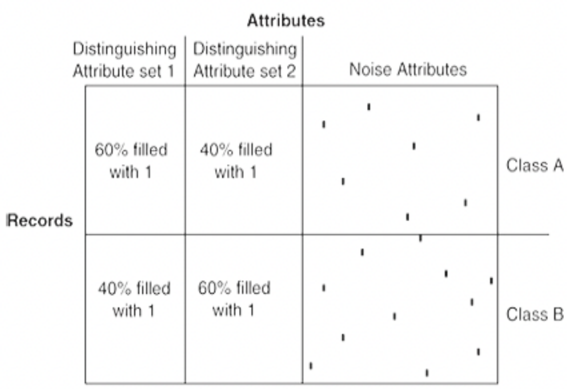
21. Given the data sets shown in Figures 4.59 below, explain how the decision tree, naïve Bayes, and k-nearest neighbor classifiers would perform on these data sets.



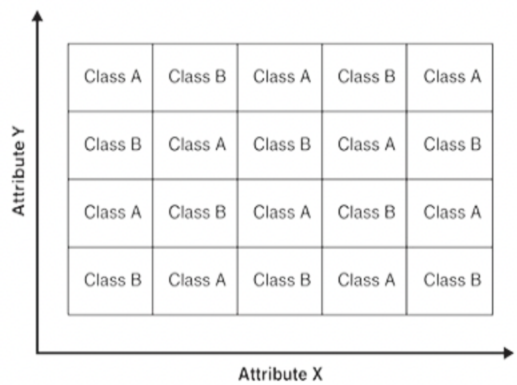

(a) Synthetic data set 1.



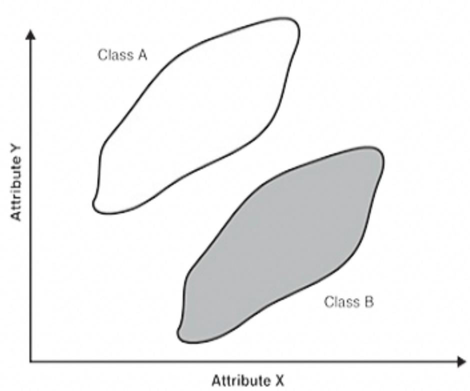

(b) Synthetic data set 2.



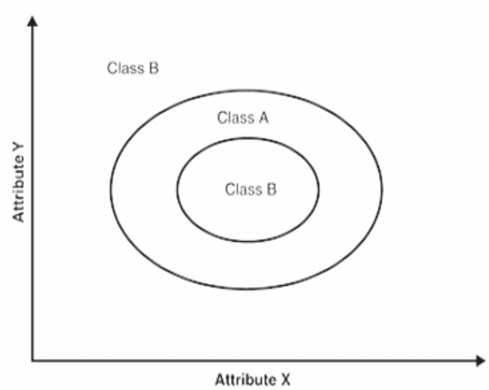

(c) Synthetic data set 3.




(d) Synthetic data set 4.




(e) Synthetic data set 5.




(f) Synthetic data set 6.

Answer:

A. K-NN classifier is sensitive to noise or many interactive attributes so it will not do well with synthetic dataset 1. Naïve Bayes classifiers are robust and can handle noise attributes because they have no impact on the probability estimates. Naïve Bayes will do well on handling synthetic dataset 1. This is the same with decision tree due to entropy gain.
B. Naïve bayes will not do well with synthetic dataset 2 because correlated attributes weaken the performance of this classifier. K-NN and decision tree can handle attributes that are dependent to each other. They will do well with this dataset.
C. Decision tree will not do well because of too many attributes to classify that can cause overfitting. K-NN will do well because it can handle interacting attributes through proximity measures that take account multiple attributes. Naïve bayes will do well too because conditional probability can be used to compare one attribute against the other.
D. Naïve Bayes will not do well because the attributes are dependent on each other. K-NN will do well because it can handle dependent attributes. Decision tree will do well since classes are binary.
E. Same reason with D, decision tree and K-NN will do well but Naïve Bayes will not do well because of dependent attributes.
F. Naïve Bayes will not do well because of dependent attributes. K-NN and decision tree will work well since they can handle attribute dependencies.

# Module 3 Assignment

Gabi Rivera

2022-11-13

# Data Science Using Python and R: Chapter 5 Hands-On Analysis

For Exercises 28–34, work with the churn data set.

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.

## Python Version

Enable python:

```
```r
library("reticulate")
```
```

Import Churn in python:

```
import pandas as pd
import numpy as np
import random
```

```
churn = pd.read_csv("churn", sep = ',')
churn.head()
```

```
##   State  Account Length  Area Code  ... CustServ Calls Old Churn  Churn
## 0    KS             128        415  ...              1    False.  False
## 1    OH             107        415  ...              1    False.  False
## 2    NJ             137        415  ...              0    False.  False
## 3    OH              84        408  ...              2    False.  False
## 4    OK              75        415  ...              3    False.  False
##
## [5 rows x 22 columns]
```

Partitioning:

```
from sklearn.model_selection import train_test_split
churn_train, churn_test = train_test_split(churn, test_size =0.33, random_state = 7)
```

Confirm partition:

```
print('Proportion of churn training instances: ', churn_train.shape[0]/churn.shape[0]
*100,
      '\nProportion of churn test instances: ', churn_test.shape[0]/churn.shape[0]*10
0)
```

```
## Proportion of churn training instances:  66.996699669967
## Proportion of churn test instances:  33.003300330033
```

# R version

Import Churn in R:

```
churn_r = read.csv("Churn", header = TRUE, sep = ",")
```

Set the seed for random number generator for later use:

```
set.seed(7)
```

Count of records in dataset:

```
n = dim(churn_r)[1]
```

Set training records to be included via random number generator (TRUE and FALSE values):

```
train_ind = runif(n) < 0.67
```

Create train and test set partitions using TRUE and FALSE values:

```
churn_trn <- churn_r[ train_ind, ]
churn_tst <- churn_r[ !train_ind, ]

head(churn_trn )
```

```
##   State Account.Length Area.Code    Phone Intl.Plan VMail.Plan VMail.Message
## 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
## 7    MA            121       510 355-9993        no        yes            24
## 9    LA            117       408 335-4719        no         no             0
##   Day.Mins Day.Calls Day.Charge Eve.Mins Eve.Calls Eve.Charge Night.Mins
## 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
## 7    218.2        88      37.09    348.5       108      29.62      212.6
## 9    184.5        97      31.37    351.6        80      29.89      215.8
##   Night.Calls Night.Charge Intl.Mins Intl.Calls Intl.Charge CustServ.Calls
## 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
## 7         118         9.57       7.5          7        2.03              3
## 9          90         9.71       8.7          4        2.35              1
##   Old.Churn Churn
## 2    False. False
## 3    False. False
## 4    False. False
## 5    False. False
## 7    False. False
## 9    False. False
```

```
head(churn_tst)
```

```
##    State Account.Length Area.Code    Phone Intl.Plan VMail.Plan VMail.Message
## 1     KS            128       415 382-4657        no        yes            25
## 6     AL            118       510 391-8027       yes         no             0
## 8     MO            147       415 329-9001       yes         no             0
## 13    IA            168       408 363-1107        no         no             0
## 19    VA             76       510 356-2992        no        yes            33
## 23    AZ            130       415 358-1958        no         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
## 6     223.4        98      37.98    220.6       101      18.75      203.9
## 8     157.0        79      26.69    103.1        94       8.76      211.8
## 13    128.8        96      21.90    104.9        71       8.92      141.1
## 19    189.7        66      32.25    212.8        65      18.09      165.7
## 23    183.0       112      31.11     72.9        99       6.20      181.8
##    Night.Calls Night.Charge Intl.Mins Intl.Calls Intl.Charge CustServ.Calls
## 1           91        11.01      10.0          3        2.70              1
## 6          118         9.18       6.3          6        1.70              0
## 8           96         9.53       7.1          6        1.92              0
## 13         128         6.35      11.2          2        3.02              1
## 19         108         7.46      10.0          5        2.70              1
## 23          78         8.18       9.5         19        2.57              0
##    Old.Churn Churn
## 1     False. False
## 6     False. False
## 8     False. False
## 13    False. False
## 19    False. False
## 23    False. False
```

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.

# Python Version

Total number of training dataset records:

```
print('Original number of records before partitioning: ', churn.shape[0],
      '\nNumber of records in Churn Training set: ', churn_train.shape[0],
      '\nNumber of records in Churn Test set: ', churn_test.shape[0])
```

```
## Original number of records before partitioning:  3333
## Number of records in Churn Training set:  2233
## Number of records in Churn Test set:  1100
```

Number of records in training dataset that are TRUE for Churn:

```
churn_train['Churn'].value_counts()
```

```
## False    1913
## True      320
## Name: Churn, dtype: int64
```

```
ratio = churn_train['Churn'].value_counts()[1] / churn_train.shape[0] * 100
ratio
```

```
## 14.330497089117777
```

# R version

Total number of training dataset records:

```
table(churn_trn$Churn)
```

```
##
## False  True
##  1902   325
```

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.

Equation:

$$x = \frac{p(records) - rare}{1 - p}$$

### Python version

```
x = ((0.2*2233) - 320) / (1-0.2)
round(x, 2)
```

```
## 158.25
```

# R version

```
a = ((0.2*(1904+317)) - 317) / (1-0.2)
a
```

```
## [1] 159
```

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.

# Python version

Isolate records to resample:

```
from random import sample
to_resample = churn_train.loc[churn_train['Churn'] ]
```

Sample from record of interest:

```
our_resample = to_resample.sample(n = 158, replace = True)
```

Rebalance training dataset with 20% of the True churn values:

```
churn_train_rebal = pd.concat([churn_train, our_resample], axis=0)
```

Confirm rebalancing results to 20% of the True churn values:

```
churn_train_rebal['Churn'].value_counts()
```

```
## False    1913
## True      478
## Name: Churn, dtype: int64
```

```
c = (478/(1913+478))*100
round(c, 0)
```

```
## 20.0
```

# R version

Isolate records to resample:

```
to.resample = which(churn_trn$Churn == "True")
```

Randomly sample from record of interest:

```
our.resample = sample(x = to.resample, size = 159, replace = TRUE)
```

Select the records whose record numbers from our.resample:

```
our.resample <- churn_trn[our.resample, ]
```

Add the resampled records back onto our original training data set:

```
trn_churn_rebal = rbind(churn_trn, our.resample)
```

Confirm rebalancing results to 20% of the True churn values:

```
t1 <- table(trn_churn_rebal$Churn)
ratio <- t1[2] / sum(t1) * 100
ratio
```

```
##   True
## 20.285
```

```
t.v1 = table(trn_churn_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          1902.0000     484.0000
## Proportion        0.7972       0.2028
```

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?

Answer: Baseline model for classification will be used to compare the model performance. So with this, the original churn attribute has about 14% true values with total of 2,233 records and 320 True values. The 14% true values can represent the All Positive Model and 86% false values represent the All Negative Model. In comparison, models developed will have to be better than 86% accuracy threshold of the baseline model to be useful.

33. Validate your partition by testing for the difference in mean day minutes for the training set versus the test set. *For a numerical variable, use the two-sample t-test for the difference in means (Larose et. al., 2019).

# Python version

Two-sample t-test of mean day minutes from the training set versus test set:

```
import scipy.stats as stats

dm_trn = churn_train_rebal[['Day Mins']].copy()
dm_tst = churn_test[['Day Mins']].copy()

print('md_trn', np.var(dm_trn),'\n' 'md_tst', np.var(dm_tst))
```

```
## md_trn Day Mins    3164.711627
## dtype: float64
## md_tst Day Mins    3010.977976
## dtype: float64
```

Usually, ratio of larger sample variance to smaller sample variance assumes equal variance at less than 4:

```
V_ratio = (3060/3011)
V_ratio
```

```
## 1.0162736632348057
```

Population have equal variance so proceed with t-test:

```
stats.ttest_ind(a=dm_trn, b=dm_tst, equal_var=True)
```

```
## Ttest_indResult(statistic=array([0.8648508]), pvalue=array([0.38718014]))
```

Result: The p-value of 0.52 is higher than significance level of 0.05. This fails to reject the null hypothesis that the two population mean are equal. This suggest that training Days Mins attribute have comparable population mean with the test Days Mins attribute.

# R version

Two-sample t-test of mean day minutes from the training set versus test set:

```
dm_train = subset(trn_churn_rebal, select = "Day.Mins")
dm_test = subset(churn_tst, select = "Day.Mins")

  t.test(dm_train, dm_test, var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  dm_train and dm_test
## t = 2.5521, df = 3490, p-value = 0.01075
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1.187340 9.059775
## sample estimates:
## mean of x mean of y
##  182.6114  177.4879
```

Result: The p-value of 0.01 is less than 0.05 alpha which mean we can reject the null hypothesis that both population means are equal. This means that the generated Days Mins training mean and test mean are not comparable.

34. Validate your partition by testing for the difference in proportion of true churn records for the training set versus the test set. *For a categorical variable with two classes, use the two-sample Z-test for the difference in proportions (Larose et. al., 2019).

## Python version

```
from statsmodels.stats.weightstats import ztest as ztest

c_trn = churn_train_rebal[['Churn']].copy()
c_tst = churn_test[['Churn']].copy()

ztest(c_trn, c_tst, value=0)
```

```
## (array([3.67370318]), array([0.00023906]))
```

Result: The p-value of 0.0002 is less than alpha 0.05 this means that churn training and test population are significantly different from each other.

# Data Science Using Python and R: Chapter 7 Hands-On Analysis

For the following exercises, work with the adult_ch6_training and adult_ch6_test data sets. Use R to solve each problem.

****Python version is done on Jupyter*****

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(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.3.6      ✔ purrr   0.3.4
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.4.1
## ✔ readr   2.1.2      ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

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

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(C50)

ad_t = read.csv("adult_ch6_training", header = TRUE, sep = ",")
colnames(ad_t)[1] = "maritalStatus"
ad_t$Income = factor(ad_t$Income)
ad_t$maritalStatus = factor(ad_t$maritalStatus)

C5 = C5.0(Income ~ maritalStatus + Cap_Gains_Losses, data = ad_t)
```

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

```
ad_tst = read.csv("adult_ch6_test", header = TRUE, sep = ",")
colnames(ad_tst)[1] = "maritalStatus"
ad_tst$Income = factor(ad_tst$Income)
ad_tst$maritalStatus = factor(ad_tst$maritalStatus)


test.X = subset(x = ad_tst, select = c("maritalStatus","Cap_Gains_Losses"))
ypred = predict(object = C5, newdata = test.X)

t1 = table(ad_tst$Income, ypred)
row.names(t1) = c("Actual: 0", "Actual: 1")

colnames(t1) = c("Predicted: 0", "Predicted: 1")
t1 = addmargins(A = t1, FUN = list(Total = sum), quiet =TRUE); t1
```

```
##              ypred
##               Predicted: 0 Predicted: 1 Total
##    Actual: 0          4658           16  4674
##    Actual: 1          1057          424  1481
##    Total              5715          440  6155
```

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.

Calculations:

```
A = round((4658+424) / 6155, 3)

Er_r = round((1- A), 3)

Sen = round(424 /1481, 3)

Sp = round(4658 / 4674, 3)

Prec = round(424 / 440, 3)

F1 = round(2*((Prec*Sen)/(Prec+Sen)), 3)

F2 = round(5*((Prec*Sen)/((4*Prec)+Sen)), 3)

F0.5 = round(1.25*((Prec*Sen)/((0.25*Prec)+Sen)), 3)
```

Model 1 Evaluation Table:

```
Table = matrix(c(A, Er_r, Sen, Sp, Prec, F1, F2, F0.5,
                 NA, NA, NA, NA, NA, NA, NA, NA), ncol=2, byrow=FALSE)

colnames(Table) = c('M1_Value', "M2_Value")
rownames(Table) = c('Accuracy', 'Error rate', 'Sensitivity', 'Specificity', 'Precisio
n', 'F1', 'F2', 'F0.5')

Table
```

```
##              M1_Value M2_Value
## Accuracy        0.826       NA
## Error rate      0.174       NA
## Sensitivity     0.286       NA
## Specificity     0.997       NA
## Precision       0.964       NA
## F1              0.441       NA
## F2              0.333       NA
## F0.5            0.654       NA
```

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

```
Accuracy_baseline = round(4674/6155, 3)
Accuracy_baseline
```

```
## [1] 0.759
```

Compared to 76% accuracy of the baseline, Model 1 is doing pretty well predicting from the observations at 83% and as a result has low error rate of 17%. Precision is high at 96% meaning that Model 1 is precise at getting correct true positive predictions. Recall or sensitivity is low at 30% which means only few true positive predictions are captured compared to the whole population. Specificity is high at 98% meaning that the actual negatives are precisely captured.For the F beta scores, ideally the higher the measured value the better as their weighted on precision and recall. In this case, these are more useful to have as a comparison between models.

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

3x Cost Matrix:

```
Table2 = matrix(c(0,4,1,0), ncol=2, byrow=TRUE)

colnames(Table2) = c('Predicted: 0', 'Predicted: 1')
rownames(Table2) = c('Actual: 0', 'Actual: 1')

Table2
```

```
##           Predicted: 0 Predicted: 1
## Actual: 0            0            4
## Actual: 1            1            0
```

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.

```
cost_C5 = matrix( c(0,1,4,0) , byrow = TRUE, ncol = 2)
cost_C5
```

```
##      [,1] [,2]
## [1,]    0    1
## [2,]    4    0
```

Rerun training dataset: M2

```
C5_costs = C5.0(Income ~ maritalStatus + Cap_Gains_Losses, data = ad_t, costs = cost_
C5)
```

```
## Warning: no dimnames were given for the cost matrix; the factor levels will be
## used
```

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.

M2 Confusion Matrix:

```
ad_tst = read.csv("adult_ch6_test", header = TRUE, sep = ",")
colnames(ad_tst)[1] = "maritalStatus"
ad_tst$Income = factor(ad_tst$Income)
ad_tst$maritalStatus = factor(ad_tst$maritalStatus)


test.X = subset(x = ad_tst, select = c("maritalStatus","Cap_Gains_Losses"))
ypred_c = predict(object = C5_costs, newdata = test.X)

t1c = table(ad_tst$Income, ypred_c)
row.names(t1c) = c("Actual: 0", "Actual: 1")

colnames(t1c) = c("Predicted: 0", "Predicted: 1")
t1c = addmargins(A = t1c, FUN = list(Total = sum), quiet =TRUE); t1c
```

```
##             ypred_c
##              Predicted: 0 Predicted: 1 Total
##   Actual: 0          4671            3  4674
##   Actual: 1          1066          415  1481
##   Total              5737          418  6155
```

Metrics calculation:

```
A2 = round((4671+415) / 6155, 3)

Er_r2 = round((1- A2), 3)

Sen2 = round(415 /1481, 3)

Sp2 = round(4671 / 4674, 3)

Prec2 = round(415 / 418, 3)

F1b = round(2*((Prec2*Sen2)/(Prec2+Sen2)), 3)

F2b = round(5*((Prec2*Sen2)/((4*Prec2)+Sen2)), 3)

F0.5b = round(1.25*((Prec2*Sen2)/((0.25*Prec2)+Sen2)), 3)
```

M1 and M2 Evaluation table:

```
Table_c = matrix(c(A, Er_r, Sen, Sp, Prec, F1, F2, F0.5,
                 A2, Er_r2, Sen2, Sp2, Prec2, F1b, F2b, F0.5b), ncol=2, byrow=FALSE)

colnames(Table_c) = c('M1_Value', "M2_Value")
rownames(Table_c) = c('Accuracy', 'Error rate', 'Sensitivity', 'Specificity', 'Precis
ion', 'F1', 'F2', 'F0.5')

Table_c
```

```
##             M1_Value M2_Value
## Accuracy       0.826    0.826
## Error rate     0.174    0.174
## Sensitivity    0.286    0.280
## Specificity    0.997    0.999
## Precision      0.964    0.993
## F1             0.441    0.437
## F2             0.333    0.327
## F0.5           0.654    0.658
```

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

Answer: Accuracy and error rate are the same for both models at 83% accuracy and 17% error rate. Both models are performing well above the baseline accuracy. Precision on getting correct true positive scores is enhanced in Model 2 at 99% compared to Model 1 at 96%. Either way both models are performing well for precision metric. Recall or sensitivity are relatively the same with Model 1 slightly higher than Model 2. The 6% difference between sensitivity in true positive predictions aginast the whole population is not much to make a definitive distinction. This is the same case with specificity at 99.7% for Model 1 and 99.9% for Model 2. Both are performing really well in capturing actual negative scores. Even with the F beta scores, there are not much overall difference that can tip in favor towards one model. Both models are performining relatively the same in my eyes.

# Data Science Using Python and R: Chapter 8 Hands-On Analysis

****Python version is done on Jupyter*****

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.

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

Import datasets:

```
library(skimr)
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

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

```
library(e1071)

fn_train = read.csv("framingham_nb_training.csv", header = TRUE, sep = ",")
fn_test = read.csv("framingham_nb_test.csv", header = TRUE, sep = ",")
```

Contingency table: Death based on education

```
ta = table(fn_train$Death, fn_train$Educ)
colnames(ta) = c("Educ = 1", "Educ = 2", "Educ = 3", "Educ = 4")
rownames(ta) = c("Death = 0", "Death = 1")
addmargins(A = ta, FUN = list(Total = sum), quiet = TRUE)
```

```
##
##               Educ = 1 Educ = 2 Educ = 3 Educ = 4 Total
##   Death = 0      173      146       84       47   450
##   Death = 1      287      135       80       48   550
##   Total          460      281      164       95  1000
```

Contingency table: Death based on sex

```
ts = table(fn_train$Death, fn_train$Sex)
colnames(ts) = c("Sex = 1", "Sex = 2")
rownames(ts) = c("Death = 0", "Death = 1")
addmargins(A = ts, FUN = list(Total = sum), quiet = TRUE)
```

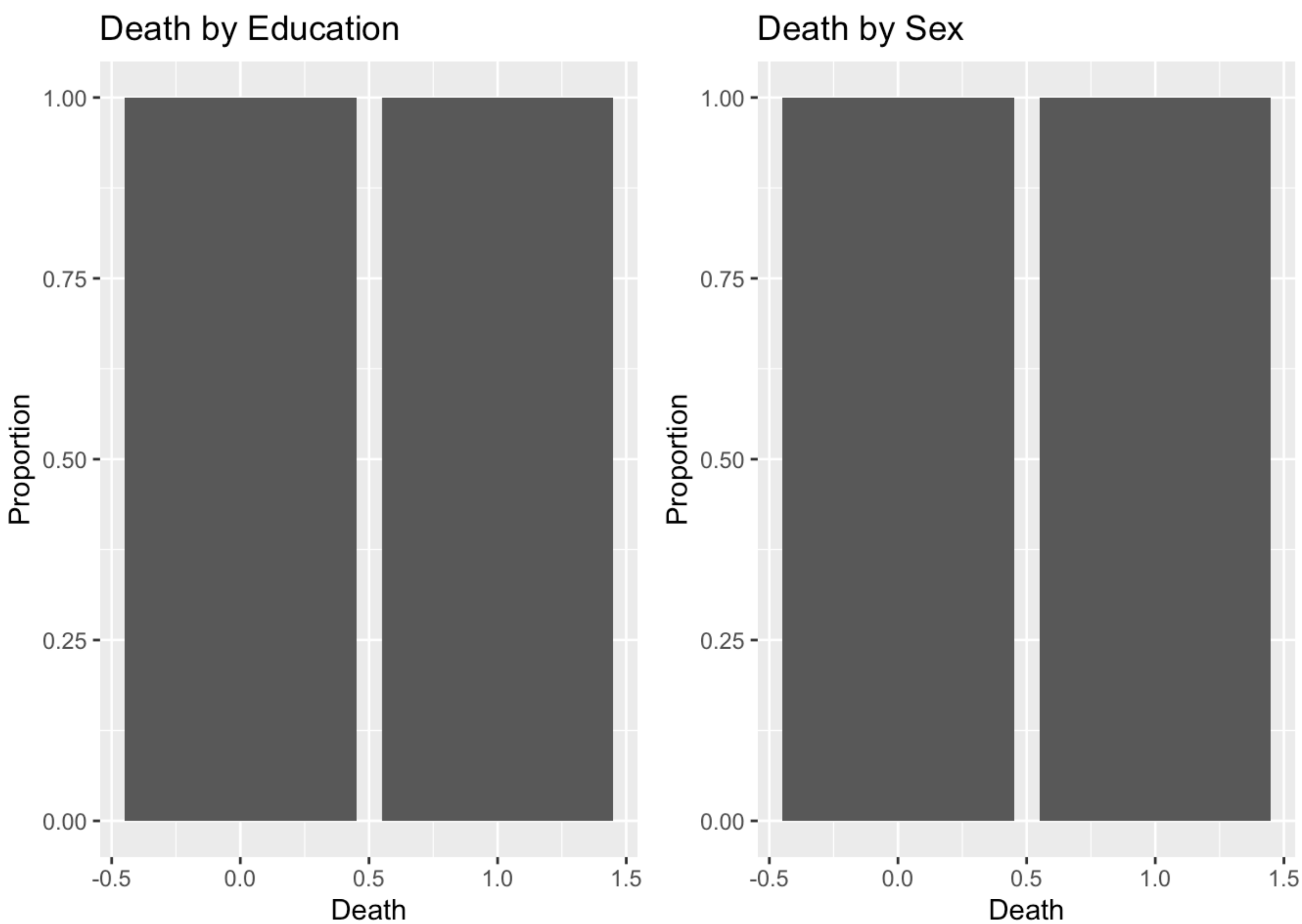
```
##
##               Sex = 1 Sex = 2 Total
##   Death = 0     184     266   450
##   Death = 1     308     242   550
##   Total         492     508  1000
```

Plot: Death based on sex and education

```
plot1 = ggplot(fn_train, aes(Death)) + geom_bar(aes(fill = Educ), position = "fill")
+ ylab('Proportion') + ggtitle("Death by Education")

plot2 = ggplot(fn_train, aes(Death)) + geom_bar( aes(fill = Sex), position = "fill")
+
ylab("Proportion")+ ggtitle("Death by Sex")

grid.arrange(plot1, plot2, nrow = 1)
```




Naive Bayes estimator:

```
nb01 = naiveBayes(formula = Death ~ Sex + Educ, data = fn_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.

Predict type of Death:

```
ypred = predict(object = nb01, newdata = fn_test)
```

Contingency table: Actual vs Predicted

```
t.preds = table(fn_test$Death, ypred)
rownames(t.preds) = c("Actual: Dead", "Actual: Living")
colnames(t.preds) = c("Predicted: Dead", "Predicted: Living")
addmargins(A = t.preds, FUN = list(Total = sum), quiet = TRUE)
```

```
##                   ypred
##                    Predicted: Dead Predicted: Living Total
##   Actual: Dead                 203               322   525
##   Actual: Living               105               370   475
##   Total                        308               692  1000
```

33. According to your table in the previous exercise, find the following values for the Naïve Bayes model:

a. Accuracy The Naive Bayes model's accuracy is 57%.

```
Accuracy_NB = ((203+370) / 1000) * 100
Accuracy_NB
```

```
## [1] 57.3
```

b. Error rate The Naive Bayes model's error rate is 43%.

```
Error_rate_NB = (100 - Accuracy_NB)
Error_rate_NB
```

```
## [1] 42.7
```

34. According to your contingency table, find the following values for the Naïve Bayes model:

a. How often it correctly classifies dead persons. The Naive Bayes model correctly classifies dead people 39% of the time.

```
Specificity_NB = (203/ 525)*100
Specificity_NB
```

```
## [1] 38.66667
```

b. How often it correctly classifies living persons. The Naive Bayes model correctly classifies living people 78% of the time.

```
Sensitivity_NB = (370/475)*100
Sensitivity_NB
```

```
## [1] 77.89474
```

# Module 3 Assignment

## Gabi Rivera || 14Nov2022 || ADS502-01

In [2]:
```
import os
os.getcwd()
```
Out[2]:
```
'/Users/gabirivera/Desktop/MSADS2/ADS502-01/Module3/Assignment'
```

In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
import statsmodels.tools.tools as stattools
```

## Data Science Using Python and R: Chapter 7 Hands-On Analysis

1. 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.

In [4]:
```
import statsmodels.tools.tools as stattools
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import tree
```

In [5]:
```
adult_tr= pd.read_csv('adult_ch6_training', sep = ',')
adult_tr.head()
```

Out[5]:

| | Marital status | Income | Cap_Gains_Losses |
|---|---|---|---|
| 0 | Never-married | <=50K | 0.02174 |
| 1 | Divorced | <=50K | 0.00000 |
| 2 | Married | <=50K | 0.00000 |
| 3 | Married | <=50K | 0.00000 |
| 4 | Married | <=50K | 0.00000 |

In [6]:
```
y = adult_tr[['Income']]

mar_np = np.array(adult_tr['Marital status'])
(mar_cat, mar_cat_dict) = stattools.categorical(mar_np, drop=True, dictnames = True)
mar_cat_pd = pd.DataFrame(mar_cat)
X = pd.concat((adult_tr[['Cap_Gains_Losses']], mar_cat_pd), axis = 1)

mar_cat_dict

X_names = ["Cap_Gains_Losses", "Divorced", "Married", "Never-married",
           "Separated", "Widowed"]

y_names = ["<=50K", ">50K"]
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tools/tools.py:1
52: FutureWarning: categorical is deprecated. Use pandas Categorical to represent catego
```

```
rical data and can get_dummies to construct dummy arrays. It will be removed after relea
se 0.13.
  warnings.warn(
```

In [7]:
```
c50_01 = DecisionTreeClassifier(criterion="entropy", max_leaf_nodes=5).fit(X,y)
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py:
1858: FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [8]:
```
c50_01.predict(X)
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py:
1858: FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[8]:
```
array(['<=50K', '<=50K', '<=50K', ..., '<=50K', '<=50K', '<=50K'],
      dtype=object)
```

# Data Science Using Python and R: Chapter 8 Hands-On Analysis

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

In [25]:
```
fn_train = pd.read_csv("framingham_nb_training.csv", sep = ',')
fn_train.head()
```

Out[25]:

| | Sex | Educ | Death |
|---|---|---|---|
| **0** | 2 | 3 | 0 |
| **1** | 2 | 2 | 0 |
| **2** | 1 | 1 | 0 |
| **3** | 2 | 1 | 0 |
| **4** | 2 | 1 | 0 |

In [26]:
```
fn_test = pd.read_csv("framingham_nb_test.csv", sep = ',')
fn_test.head()
```

Out[26]:

| | Sex | Educ | Death |
|---|---|---|---|
| **0** | 1 | 1 | 0 |
| **1** | 1 | 2 | 0 |
| **2** | 1 | 3 | 0 |
| **3** | 1 | 1 | 0 |
| **4** | 2 | 2 | 0 |

Contingency table: Death based on sex

In [30]:
```
t1 = pd.crosstab(fn_train['Death'], fn_train['Sex'])
t1['Total'] = t1.sum(axis=1)
t1.loc['Total'] = t1.sum()
t1
```

Out[30]:

| Sex | 1 | 2 | Total |
|---|---|---|---|

| Death | | | |
|---|---|---|---|
| 0 | 184 | 266 | 450 |
| 1 | 308 | 242 | 550 |
| Total | 492 | 508 | 1000 |

Contingency table: Death based on sex

In [31]:
```
t2 = pd.crosstab(fn_train['Death'], fn_train['Educ'])
t2['Total'] = t2.sum(axis=1)
t2.loc['Total'] = t2.sum()
t2
```

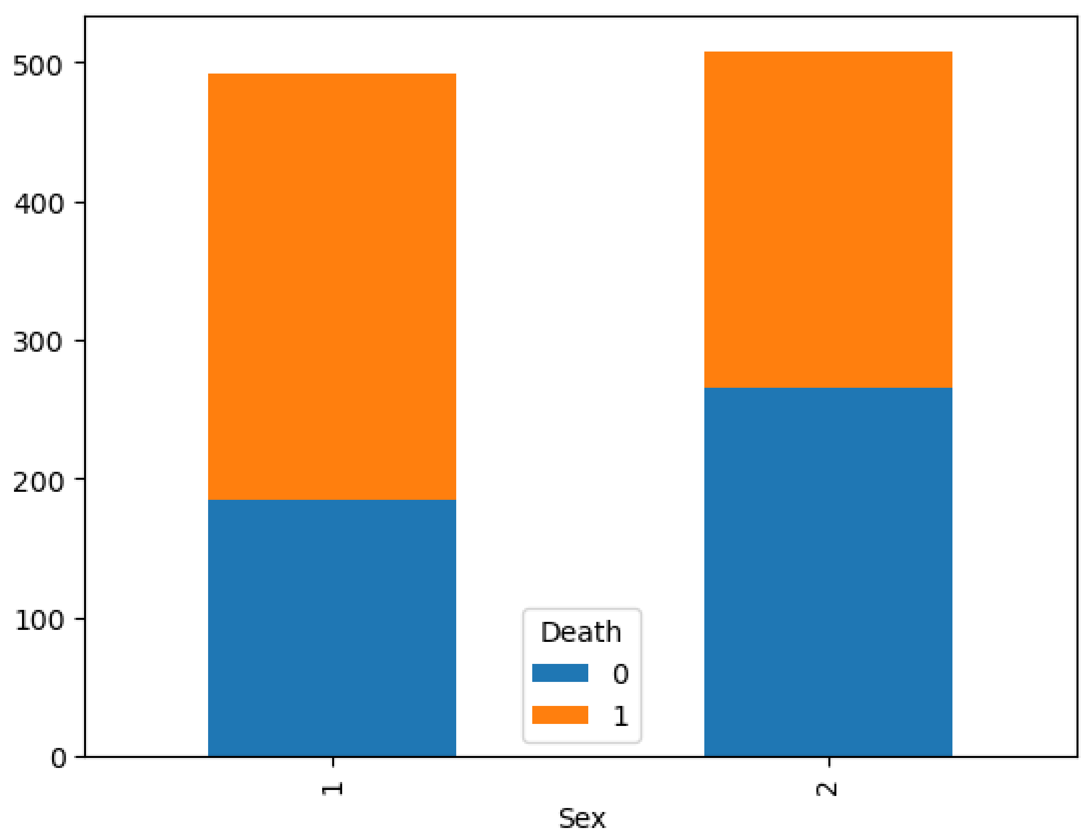
Out[31]:

| Educ | 1 | 2 | 3 | 4 | Total |
|---|---|---|---|---|---|
| **Death** | | | | | |
| 0 | 173 | 146 | 84 | 47 | 450 |
| 1 | 287 | 135 | 80 | 48 | 550 |
| Total | 460 | 281 | 164 | 95 | 1000 |

Plot: Death based on sex

In [29]:
```
t1_plot = pd.crosstab(fn_train['Sex'], fn_train['Death'])
t1_plot.plot(kind='bar', stacked = True)
```

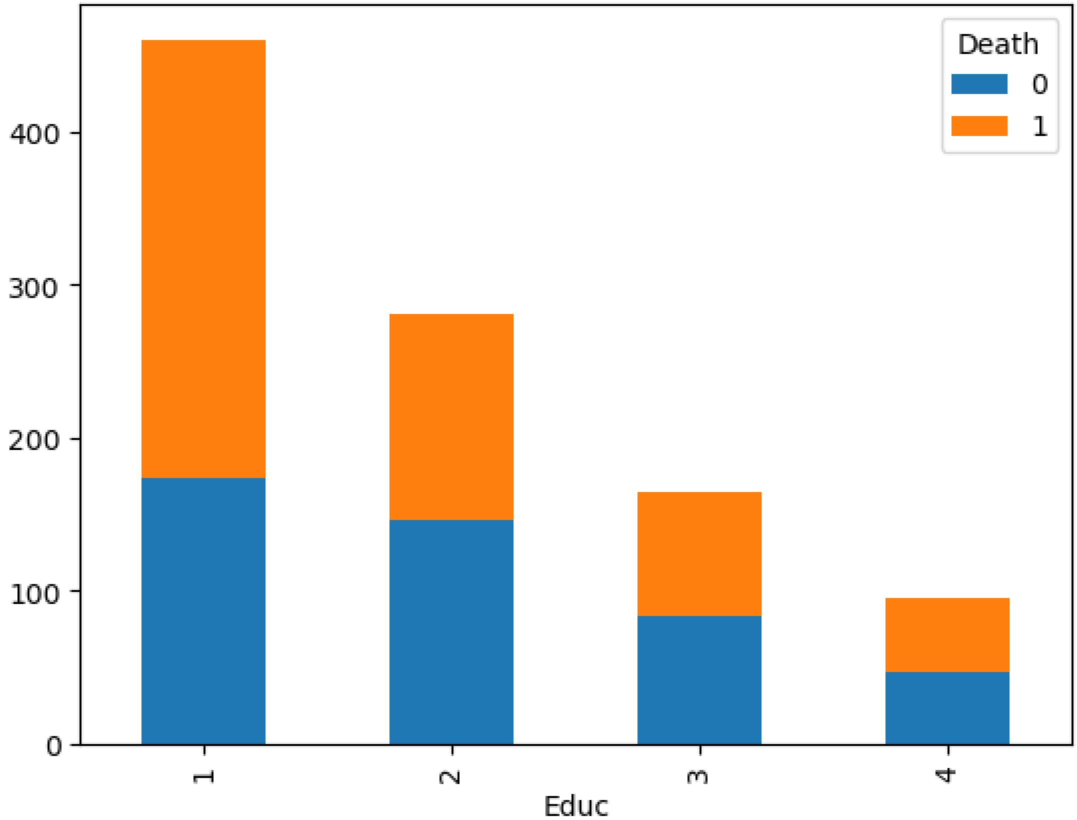
Out[29]: <AxesSubplot:xlabel='Sex'>




In [ ]:
```
Plot: Death based on education
```

In [33]:
```
t2_plot = pd.crosstab(fn_train['Educ'], fn_train['Death'])
t2_plot.plot(kind='bar', stacked = True)
```

Out[33]: <AxesSubplot:xlabel='Educ'>




Naive Bayes dataset prep:

In [42]:
```
X_Sex_ind = np.array(fn_train['Sex'])
(X_Sex_ind , X_Sex_ind_dict) = stattools.categorical(X_Sex_ind,drop=True, dictnames = Tr
X_Sex_ind = pd.DataFrame(X_Sex_ind)

X_Educ_ind = np.array(fn_train['Educ'])
(X_Educ_ind , X_Educ_ind_dict) = stattools.categorical(X_Educ_ind, drop=True, dictnames
X_Educ_ind = pd.DataFrame(X_Educ_ind)



X = pd.concat((X_Sex_ind, X_Educ_ind), axis = 1)
Y = fn_train['Death']
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tools/tools.py:1
52: FutureWarning: categorical is deprecated. Use pandas Categorical to represent catego
rical data and can get_dummies to construct dummy arrays. It will be removed after relea
se 0.13.
  warnings.warn(
```

In [51]:
```
nb_01 = MultinomialNB().fit(X, Y)
```

1. 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.

Naïve Bayes model on the framingham_nb_test data set:

In [43]:
```
X_Sex_ind_test = np.array(fn_test['Sex'])
(X_Sex_ind_test, X_Sex_ind_dict_test) = stattools.categorical(X_Sex_ind_test,drop=True,
X_Sex_ind_test = pd.DataFrame(X_Sex_ind_test)
```

```
X_Educ_ind_test = np.array(fn_test['Educ'])
(X_Educ_ind_test, X_Educ_ind_dict_test) = stattools.categorical(X_Educ_ind_test, drop=Tr
X_Educ_ind_test = pd.DataFrame(X_Educ_ind_test)

X_test = pd.concat((X_Sex_ind_test, X_Educ_ind_test), axis = 1)
Y_predicted = nb_01.predict(X_test)
```

Naive Bayes contingency table:

In [49]:
```
ypred = pd.crosstab(fn_test['Death'], Y_predicted, rownames = ['Actual'], colnames = ['P

ypred['Total'] = ypred.sum(axis=1); ypred.loc['Total'] = ypred.sum(); ypred
```

Out[49]:

| Predicted | 0 | 1 | Total |
|---|---|---|---|
| **Actual** | | | |
| **0** | 203 | 322 | 525 |
| **1** | 105 | 370 | 475 |
| **Total** | 308 | 692 | 1000 |

1. According to your table in the previous exercise, find the following values for the Naïve Bayes model:

a. Accuracy

In [56]:
```
Accuracy_NB = ((203+370) / 1000) * 100
Accuracy_NB
```

Out[56]:
```
57.3
```

b. Error rate

In [57]:
```
Error_rate_NB = (100 - Accuracy_NB)
Error_rate_NB
```

Out[57]:
```
42.7
```

1. According to your contingency table, find the following values for the Naïve Bayes model:

a. How often it correctly classifies dead persons.

In [61]:
```
Specificity_NB = (203/ 525)*100
round(Specificity_NB, 1)
```

Out[61]:
```
38.7
```

b. How often it correctly classifies living persons.

In [63]:
```
Sensitivity_NB = (370/475)*100
round(Sensitivity_NB, 1)
```

Out[63]:
```
77.9
```