Classification

Sovanna Ramirez & Sanjana Jadhav

2023-02-17

In this notebook, we will be showing some models for Classification: Logistic Regression Model and Naive Bayes. Logistic Regression models relationships between one response variable and predictor variables. Naive Bayes assumes all predictors are independent and determines the conditional probability of each category of a predictor. Naive Bayes has higher bias and lower variance than Logistic Regression and you will see what this means as you read through this notebook.

Data Exploration

This example looks at the data set **UCI Adult Income** as an intro into Logistic Regression & Naive Bayes. The data set was downloaded from here: https://www.kaggle.com/datasets/wenruliu/adult-income-dataset

The "read.csv" function takes a file path as input and loads the contents of the file into a data frame named "df."

```
df <- read.csv("adult.csv")
```

Data Cleaning Here, we are using "sapply()" to apply a function to the entire data frame, "df."
* The anonymous function "function(x)" uses the "sum()" and is.na()" functions to find the amount of missing values in a column.
* A vector containing the missing values for all the columns in "df" is displayed below.

```
sapply(df, function(x) sum(is.na(x)==TRUE))
```

```
##            age      workclass         fnlwgt      education educational.num
##              0              0              0              0               0
##  marital.status     occupation   relationship           race          gender
##              0              0              0              0               0
##    capital.gain   capital.loss hours.per.week native.country          income
##              0              0              0              0               0
```

Thankfully, the data does not have any missing values.

str() Function

The "str()" function displays the structure of the data frame. This helps us find the data types of each of the columns.

```
str(df)
```

```
## 'data.frame':    48842 obs. of  15 variables:
##  $ age            : int  25 38 28 44 18 34 29 63 24 55 ...
##  $ workclass      : chr  "Private" "Private" "Local-gov" "Private" ...
##  $ fnlwgt         : int  226802 89814 336951 160323 103497 198693 227026 104626 369667 104996 ...
##  $ education      : chr  "11th" "HS-grad" "Assoc-acdm" "Some-college" ...
##  $ educational.num: int  7 9 12 10 10 6 9 15 10 4 ...
##  $ marital.status : chr  "Never-married" "Married-civ-spouse" "Married-civ-spouse" "Married-civ-spous
##  $ occupation     : chr  "Machine-op-inspct" "Farming-fishing" "Protective-serv" "Machine-op-inspct"
##  $ relationship   : chr  "Own-child" "Husband" "Husband" "Husband" ...
##  $ race           : chr  "Black" "White" "White" "Black" ...
##  $ gender         : chr  "Male" "Male" "Male" "Male" ...
##  $ capital.gain   : int  0 0 0 7688 0 0 0 3103 0 0 ...
##  $ capital.loss   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hours.per.week : int  40 50 40 40 30 30 40 32 40 10 ...
##  $ native.country : chr  "United-States" "United-States" "United-States" "United-States" ...
##  $ income         : chr  "<=50K" "<=50K" ">50K" ">50K" ...
```

factor() Function

The "as.factor()" function is used to convert a column's data type to a factor variable. This way, it is easier to represent categories. For this example, here are the variables that would have their individual categories:
* marital.status
* income (<= 50k or > 50k)
* race
* gender
* occupation

-c() Function

The following columns will be deleted as the data frame contains overlap/irrelevant information that may affect the accuracy.
* workclass
* fnlwgt
* education
* relationship
* capital.gain
* capital.loss
* native.country

We also use str() to view the current data frame.

```r
df$marital.status <- as.factor(df$marital.status)
df$income <- as.factor(df$income)
df$race <- as.factor(df$race)
df$gender <- as.factor(df$gender)
df$occupation <- as.factor(df$occupation)
df <- df[-c(2:4,8,11,12,14)]
str(df)
```

```
## 'data.frame':    48842 obs. of  8 variables:
##  $ age            : int  25 38 28 44 18 34 29 63 24 55 ...
##  $ educational.num: int  7 9 12 10 10 6 9 15 10 4 ...
##  $ marital.status : Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 5 3 3 3 5 5 5 3 5 3 ...
```

```
## $ occupation     : Factor w/ 15 levels "?","Adm-clerical",..: 8 6 12 8 1 9 1 11 9 4 ...
## $ race           : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 3 5 5 3 5 5 3 5 5 5 ...
## $ gender         : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 2 2 2 1 2 ...
## $ hours.per.week : int  40 50 40 40 30 30 40 32 40 10 ...
## $ income         : Factor w/ 2 levels "<=50K",">50K": 1 1 2 2 1 1 1 2 1 1 ...
```

Divide into Train/Test (80/20)

- set.seed(1234): ensures that the train/test data is the same each time the code is run
- 80% of the data is used to train the model and 20% of the test the model
- replace=FALSE ensures that there is no overlap of the data in train/test

```
set.seed(1234)
i <- sample(1:nrow(df), nrow(df)*0.8,replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

summary() Function

- The "summary()" is used to display statistics about the train data frame, consisting of values such as max/min for numerical data types and the number of occurrences of each category for factor variables.
- the "str()" is called again to visualize the structure of the train data.

```
summary(train)
```

```
##       age        educational.num               marital.status
##  Min.   :17.00   Min.   : 1.00   Divorced             : 5301
##  1st Qu.:28.00   1st Qu.: 9.00   Married-AF-spouse    :   30
##  Median :37.00   Median :10.00   Married-civ-spouse   :17897
##  Mean   :38.64   Mean   :10.06   Married-spouse-absent:  505
##  3rd Qu.:48.00   3rd Qu.:12.00   Never-married        :12891
##  Max.   :90.00   Max.   :16.00   Separated            : 1240
##                                  Widowed              : 1209
##            occupation                      race            gender
##  Prof-specialty : 4901   Amer-Indian-Eskimo:  373   Female:12970
##  Craft-repair   : 4885   Asian-Pac-Islander: 1226   Male  :26103
##  Exec-managerial: 4827   Black             : 3771
##  Adm-clerical   : 4465   Other             :  327
##  Sales          : 4423   White             :33376
##  Other-service  : 3961
##  (Other)        :11611
##  hours.per.week    income
##  Min.   : 1.00   <=50K:29812
##  1st Qu.:40.00   >50K : 9261
##  Median :40.00
##  Mean   :40.37
##  3rd Qu.:45.00
##  Max.   :99.00
##
```

table() Function

The "table()" function is used to view the different categories that occur in a vector and their frequencies. An example is shown below for the occupation column.

```
table(train$occupation)
```

```
##
##                 ?      Adm-clerical      Armed-Forces      Craft-repair
##              2259              4465                14              4885
##   Exec-managerial   Farming-fishing Handlers-cleaners Machine-op-inspct
##              4827              1200              1672              2424
##     Other-service   Priv-house-serv    Prof-specialty   Protective-serv
##              3961               202              4901               781
##             Sales      Tech-support  Transport-moving
##              4423              1165              1894
```

head() and tail() Functions

- head(): shows the first few rows the train data frame
- tail(): shows the last few rows the train data frame

```
head(train)
```

```
##       age educational.num        marital.status      occupation  race gender
## 40784  45              13         Never-married  Prof-specialty White   Male
## 40854  28               8    Married-civ-spouse  Prof-specialty White   Male
## 41964  30               5    Married-civ-spouse   Other-service White   Male
## 15241  30               7    Married-civ-spouse    Craft-repair White   Male
## 33702  57              12              Divorced  Prof-specialty Black Female
## 35716  67              14 Married-spouse-absent Exec-managerial White   Male
##       hours.per.week income
## 40784             40  <=50K
## 40854             45  <=50K
## 41964             37  <=50K
## 15241             40  <=50K
## 33702             40  <=50K
## 35716             55   >50K
```

```
tail(train)
```

```
##       age educational.num     marital.status     occupation  race gender
## 763    36              13      Never-married Prof-specialty White Female
## 39524  20               9      Never-married          Sales White   Male
## 27799  23              13      Never-married   Tech-support White Female
## 2000   32              10      Never-married  Other-service White   Male
## 36270  38              13 Married-civ-spouse          Sales White   Male
## 17770  57               9           Divorced   Adm-clerical White Female
##       hours.per.week income
## 763               40   >50K
## 39524             48  <=50K
```

```
## 27799                  20   <=50K
## 2000                   34   <=50K
## 36270                  55    >50K
## 17770                  40   <=50K
```

cor() Function

The "cor()" function computes the correlation between two variables in a data frame. For example, the code below calculates the correlation between "hours.per.week" and "age." As we can say, the correlation is very close to 0. There is no correlation.

```
cor(train$hours.per.week, train$age)
```

```
## [1] 0.0696672
```

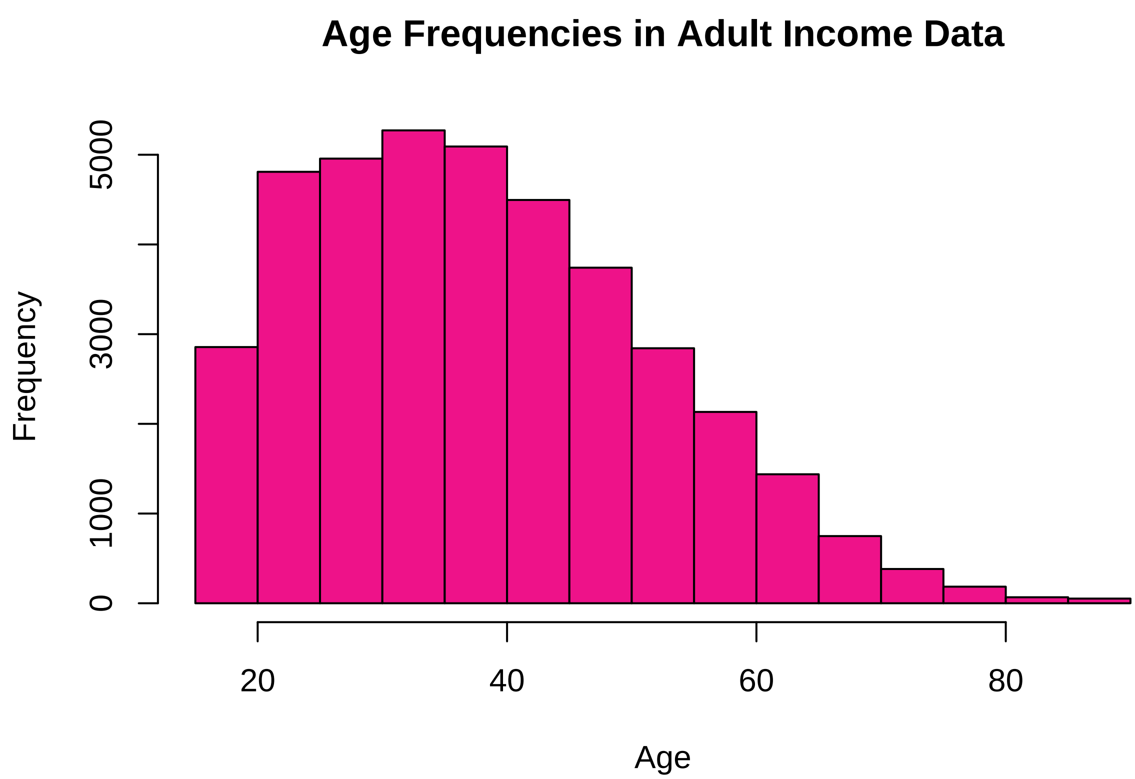
Data Visualization

Data Visualization helps us find patterns in the data.

hist() Function

For instance, this is a **histogram** that shows the frequency of the different ages in the train data.

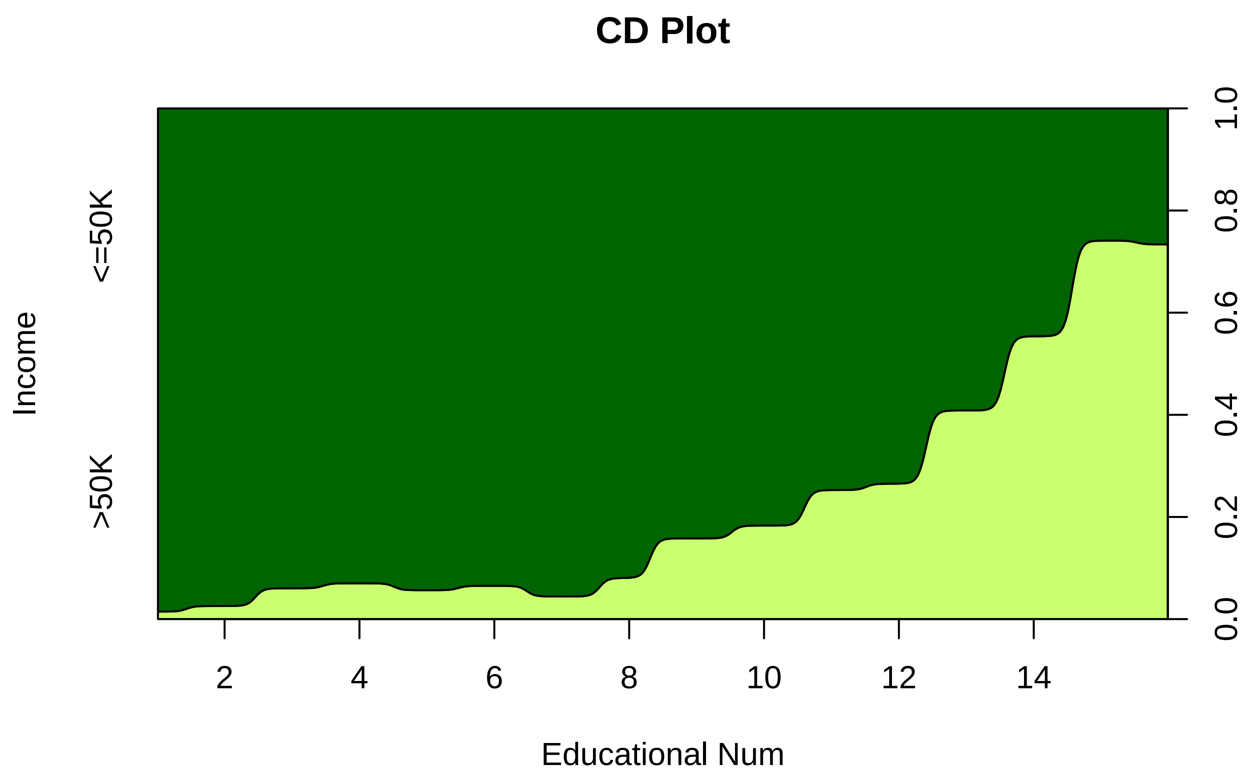
```
hist(train$age, col="deeppink2", main="Age Frequencies in Adult Income Data", xlab="Age")
```



cdplot() Function

The "cdplot()" function displays the conditional density, which shows us how a numerical value affects categorical data. For instance, the code below shows us how Education Num affects Income.

```
cdplot(train$educational.num, train$income, col=c("darkolivegreen1","darkgreen"),
       xlab="Educational Num", ylab="Income", main="CD Plot")
```



Logistic Regression Model

In the code below, we are creating a logistic regression model using the train data.
* glm(): generalized linear function used for logistic regression
* income~.: all the other variables in the train data frame and predictors used to predict "income"
* data=train: we are using the train data frame
* family="binomial": a binomial logistic regression model is used as the income variable only has 2 levels (<= 50k or > 50k)

```
glm1 <- glm(income~., data=train, family="binomial")
summary(glm1)
```

```
##
## Call:
## glm(formula = income ~ ., family = "binomial", data = train)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7162  -0.5566  -0.2388  -0.0621   3.5999
##
## Coefficients:
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                         -9.336638   0.245438 -38.041  < 2e-16 ***
## age                                  0.027844   0.001357  20.520  < 2e-16 ***
## educational.num                      0.294629   0.007837  37.594  < 2e-16 ***
## marital.statusMarried-AF-spouse      2.608351   0.440994   5.915 3.32e-09 ***
## marital.statusMarried-civ-spouse     2.120707   0.056659  37.429  < 2e-16 ***
## marital.statusMarried-spouse-absent  0.209732   0.176796   1.186  0.23551
## marital.statusNever-married         -0.407159   0.068987  -5.902 3.59e-09 ***
## marital.statusSeparated             -0.078956   0.134807  -0.586  0.55808
## marital.statusWidowed               -0.048735   0.126369  -0.386  0.69975
## occupationAdm-clerical               0.764321   0.099985   7.644 2.10e-14 ***
## occupationArmed-Forces               1.588105   0.751357   2.114  0.03455 *
## occupationCraft-repair               0.667622   0.095458   6.994 2.67e-12 ***
## occupationExec-managerial            1.436350   0.093757  15.320  < 2e-16 ***
## occupationFarming-fishing           -0.633190   0.135521  -4.672 2.98e-06 ***
## occupationHandlers-cleaners         -0.100134   0.138664  -0.722  0.47021
## occupationMachine-op-inspct          0.340153   0.110119   3.089  0.00201 **
## occupationOther-service             -0.349668   0.122678  -2.850  0.00437 **
## occupationPriv-house-serv           -1.128037   0.654793  -1.723  0.08494 .
## occupationProf-specialty             1.230336   0.095439  12.891  < 2e-16 ***
## occupationProtective-serv            0.904338   0.126164   7.168 7.61e-13 ***
## occupationSales                      0.894369   0.096295   9.288  < 2e-16 ***
## occupationTech-support               1.147029   0.116882   9.814  < 2e-16 ***
## occupationTransport-moving           0.512673   0.108360   4.731 2.23e-06 ***
## raceAsian-Pac-Islander               0.399163   0.205725   1.940  0.05235 .
## raceBlack                            0.302891   0.197294   1.535  0.12473
## raceOther                            0.258159   0.278688   0.926  0.35427
## raceWhite                            0.524575   0.188245   2.787  0.00533 **
## genderMale                           0.143841   0.044167   3.257  0.00113 **
## hours.per.week                       0.030021   0.001364  22.015  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 42794  on 39072  degrees of freedom
## Residual deviance: 27720  on 39044  degrees of freedom
## AIC: 27778
##
## Number of Fisher Scoring iterations: 6
```

The purpose of this logistic regression model is to predict the probability an adult makes an income of over 50K, given other predictors such as age, education level, marital status, occupation, race, gender, and hours of work per week. The summary shows the following values:
* regression coefficient: This shows the coefficient in log odds for each of the predictors in the train data frame. In our data, "Being Married," and working in "Prof-speciality" or "Tech-support" has a higher probability of making an income of about 50k.
* standard error: the average space between the observations and the regression line
* z-value: regression coefficient/standard error (tells us how many far we are away from the mean and it can

be positive or negative)
* p-value: indicates significance and if the value is less than 0.05, the predictor strongly influences the model. In our data, being a white male strongly influences the model

Predict using the Test Data

- accuracy: number of correct predictions divided by all predictions
- confusion matrix:

  TN (correct <=50k), FP (incorrect > 50k), FN (incorrect <=50k), TP (correct >50k) –> since most of the values are on the diagonal, they are equal to their true values. therefore, this model is a good fit

```
probs <- predict(glm1, newdata=test, type="response")
pred <- ifelse(probs>0.5, ">50K", "<=50K")
acc <- mean(pred==test$income)
print(paste("accuracy = ", acc))
```

```
## [1] "accuracy =  0.82567304739482"
```

```
table(pred, test$income)
```

```
##
## pred    <=50K >50K
##   <=50K  6769 1129
##   >50K    574 1297
```

Find Sensitivity, Specificity, and Kappa

- sensitivity: the model correctly predicts 92.18% of positive case
- specificity: the model correctly only predicts 53.46% of negative case
- kappa: 0.4943 –> accounts for correct prediction by chance

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

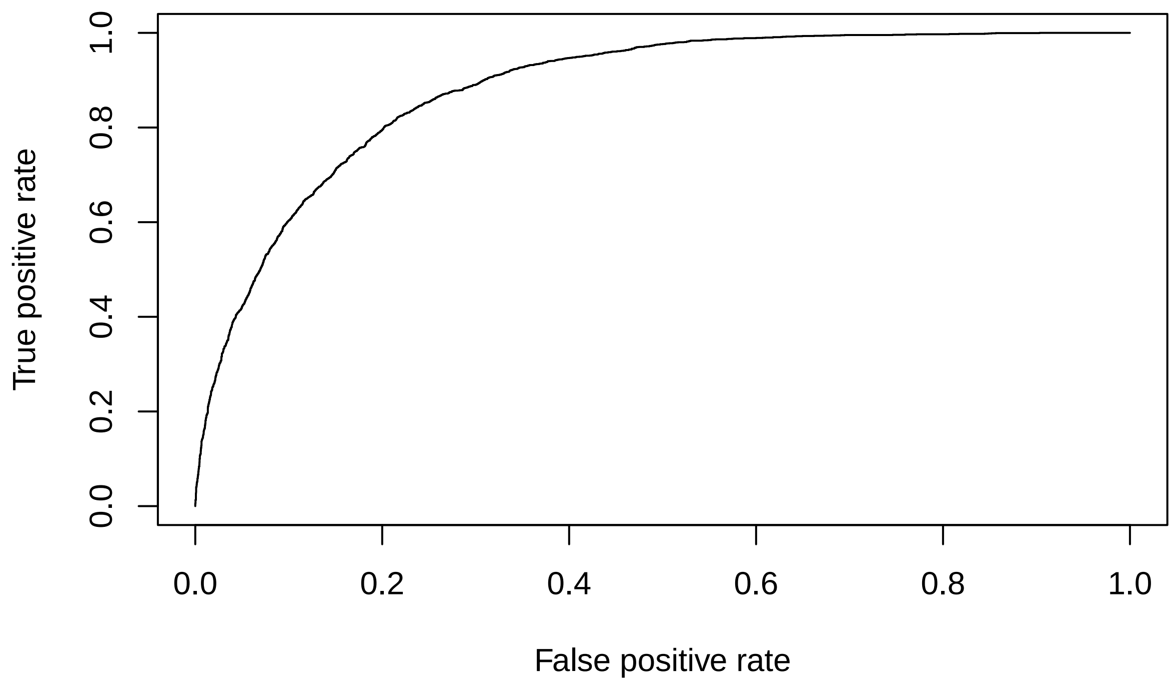
```
confusionMatrix(as.factor(pred), reference=test$income)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K >50K
##      <=50K  6769 1129
##      >50K    574 1297
##
##                Accuracy : 0.8257
##                  95% CI : (0.818, 0.8331)
```

```
##      No Information Rate : 0.7517
##      P-Value [Acc > NIR] : < 2.2e-16
##
##                    Kappa : 0.4943
##
##   Mcnemar's Test P-Value : < 2.2e-16
##
##              Sensitivity : 0.9218
##              Specificity : 0.5346
##           Pos Pred Value : 0.8571
##           Neg Pred Value : 0.6932
##               Prevalence : 0.7517
##           Detection Rate : 0.6929
##     Detection Prevalence : 0.8085
##        Balanced Accuracy : 0.7282
##
##         'Positive' Class : <=50K
##
```

ROC and AUC

```
library(ROCR)
pr <- prediction(probs, test$income)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```




```r
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.8810923
```

- ROC: plots the sensitivity against the specificity
- AUC: 0.8810923 –> this is the area under the curve and a value close to 1 is better.

Naive Bayes

- The A-priori probabilities are also displayed for income: <= 50k: 0.762981 and >50k: 0.2370179. These are baseline probabilities.
- This model also displays the independent conditional probability for each predictor, and each predictor is independent of another.

```r
library(e1071)
nb1 <- naiveBayes(income~., data=train)
nb1
```

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

```
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##     <=50K      >50K
## 0.7629821 0.2370179
##
## Conditional probabilities:
##        age
## Y           [,1]     [,2]
##   <=50K 36.90403 14.08109
##   >50K  44.20862 10.55107
##
##        educational.num
## Y            [,1]     [,2]
##   <=50K  9.586509 2.437105
##   >50K  11.599611 2.396753
##
##        marital.status
## Y           Divorced Married-AF-spouse Married-civ-spouse Married-spouse-absent
##   <=50K 0.1603045753      0.0006373273       0.3345968067          0.0153629411
##   >50K  0.0563654033      0.0011877767       0.8554151819          0.0050750459
##        marital.status
## Y       Never-married    Separated      Widowed
##   <=50K  0.4127532537 0.0390111365 0.0373339595
##   >50K   0.0632761041 0.0083144369 0.0103660512
##
##        occupation
## Y                  ? Adm-clerical Armed-Forces Craft-repair Exec-managerial
##   <=50K 0.0686636254 0.1289413659 0.0003018919 0.1273983631    0.0849993291
##   >50K  0.0228916964 0.0670553936 0.0005398985 0.1173739337    0.2475974517
##        occupation
## Y       Farming-fishing Handlers-cleaners Machine-op-inspct Other-service
##   <=50K    0.0358580437      0.0524620958      0.0711793908  0.1275660808
##   >50K     0.0141453407      0.0116618076      0.0326098693  0.0170607926
##        occupation
## Y       Priv-house-serv Prof-specialty Protective-serv        Sales
##   <=50K    0.0066751644   0.0898966859    0.0184489467 0.1089494163
##   >50K     0.0003239391   0.2398229133    0.0249433107 0.1268761473
##        occupation
## Y       Tech-support Transport-moving
##   <=50K 0.0281430297     0.0505165705
##   >50K  0.0352013821     0.0418961235
##
##        race
## Y       Amer-Indian-Eskimo Asian-Pac-Islander       Black       Other
##   <=50K        0.011102912        0.029853750 0.111699987 0.009660539
##   >50K         0.004535147        0.036281179 0.047619048 0.004211208
##        race
## Y             White
##   <=50K 0.837682812
##   >50K  0.907353418
##
##        gender
```

```
## Y          Female      Male
##   <=50K 0.3875621 0.6124379
##   >50K  0.1528993 0.8471007
##
##        hours.per.week
## Y           [,1]     [,2]
##   <=50K 38.81038 12.34431
##   >50K  45.40654 11.05481
```

Predict using the Test Data

- confusion matrix:

  TN (correct <=50k), FP (incorrect > 50k), FN (incorrect <=50k), TP (correct >50k) –> since most of the values are on the diagonal, they are equal to their true values. therefore, this model is a good fit
- mean: number of correct predictions divided by all predictions

```
pred2 <- predict(nb1, newdata = test, type="class")
table(pred2, test$income)
```

```
##
## pred2   <=50K >50K
##   <=50K  6433  808
##   >50K    910 1618
```

```
mean(pred2 == test$income)
```

```
## [1] 0.8241376
```

Both models only have a very slight difference in their accuracies.
* Logistic Regression: 0.82567304739482
* Naive Bayes: 0.8241376

Strengths and Weaknesses of Logistic Regression and Naive Bayes

Both methods can handle numeric and categorical data. Logistic Regression does better on larger data whereas Naive Bayes does better on smaller data. Naive Bayes has a lower variance than Logistic Regression. This is a drawback of Naive Bayes as predictors are not always independent of each other. However, this is a strength of Logistic Regression as it can find relationships between predictors. Although, a drawback of Logistic Regression is that it tends to overfit. This usually occurs when there are too many predictors and the model tries to satisfy each relationship, rather than trying to find the underlying trends. When it comes to choosing one method over the other, it is best to use both and see how the values differ.

Strengths and Weaknesses of Classification Metrics

The accuracy is an easy and necessary metric used to determine if the model is a good fit. Kappa is used to ensure that correctness by chance is factored in. Sensitivity and Specificity are used to determine correct positive and negative values respectively. The ROC measures how the specificity and sensitivity are related to each other, and the AUC is the area under the curve. The major drawback of using these metrics is that skewed data will result in incorrect values.