LAU8-By-Shantal-Cruz

```
# Calculate the minimum and maximum incomes in the city assigned to you.
library(readr)
data <- readr::read_csv("4063Midterm.csv")
```

```
## Rows: 1000 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Fname, Lname, gender, City
## dbl (9): ID, FamilyIncome, EdYears, FamilySize, Grocery, Cosmatics, MF, Boug...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# 1) Divide the data set to Train, Test Data Sets

# Installing Packages
# install.packages("e1071")
# install.packages("caTools")
# install.packages("caret")
# install.packages("plotROC")

# Loading package
library(e1071)
library(caTools)
library(caret)
```

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

```
library(class)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(ROCR)

# # Splitting data into train
# # and test data
# split <- sample.split(data, SplitRatio = 0.7)
# train_cl <- subset(data, split == "TRUE")
# test_cl <- subset(data, split == "FALSE")

# randomize data
```

```r
set.seed(123)
data <- data[sample(nrow(data)),] # shuffle

# convert string data to numerical factors
# data$gender <- as.numeric(factor(data$gender))
data$City <- as.numeric(factor(data$City))
data$Fname <- as.numeric(factor(data$Fname))
data$Lname <- as.numeric(factor(data$Lname))

# remove ID and gender columns
x <- data[,c(-1, -4, -11)]
y <- data[4]

# 70% of the sample size
smp_size <- floor(0.7 * nrow(data))

train_ind <- sample(seq_len(nrow(data)), size = smp_size)

x_train <- x[train_ind, ]
x_test <- x[-train_ind, ]

y_train <- y[train_ind, ]
y_test <- y[-train_ind, ]

# 2) Use Naive Bayes Classifier for predicting whether the customer was male or female

# Fitting Naive Bayes Model
# to training dataset
nb <- naiveBayes(x = x_train, y = y_train)
nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = x_train, y = y_train)
##
## A-priori probabilities:
## y_train
##         F         M
## 0.4885714 0.5114286
##
## Conditional probabilities:
##        Fname
## y_train     [,1]     [,2]
##       F 258.3684 156.6620
##       M 282.6313 157.6764
##
##        Lname
## y_train     [,1]     [,2]
##       F 455.7924 260.0456
##       M 432.8687 254.1149
##
##        City
```

```
## y_train     [,1]     [,2]
##       F 3.880117 1.483126
##       M 3.776536 1.422235
##
##        FamilyIncome
## y_train     [,1]     [,2]
##       F 29279.84 10437.56
##       M 30182.38 11445.54
##
##        EdYears
## y_train     [,1]     [,2]
##       F 14.04386 6.697122
##       M 14.48603 7.370056
##
##        FamilySize
## y_train     [,1]     [,2]
##       F 5.125731 1.210406
##       M 5.053073 1.247958
##
##        Grocery
## y_train     [,1]     [,2]
##       F 1092.442 192.7788
##       M 1098.866 190.3706
##
##        Cosmatics
## y_train     [,1]     [,2]
##       F 452.0848 69.48462
##       M 453.0196 75.61396
##
##        BoughtCosmatics
## y_train      [,1]       [,2]
##       F 0.6669850 0.09995212
##       M 0.3224185 0.10940031
##
##        Response
## y_train      [,1]       [,2]
##       F 0.6669850 0.09995212
##       M 0.3224185 0.10940031
```

```r
# Predicting on test data
y_pred_nb <- predict(nb, newdata = x_test)

# getting probability of males for ROC
y_pred_nb_probM <- predict(nb, newdata = x_test, "raw")[,2]
y_pred_nb_probM
```

```
##   [1] 9.997879e-01 2.496434e-05 2.769226e-06 9.999994e-01 9.999998e-01
##   [6] 1.000000e+00 9.997452e-01 9.967862e-01 2.537807e-07 1.979078e-08
##  [11] 9.999999e-01 8.991950e-05 2.382451e-05 9.670634e-01 3.608832e-05
##  [16] 9.978119e-01 2.640571e-01 9.999990e-01 3.984744e-09 2.677091e-08
##  [21] 9.999913e-01 2.011480e-05 9.998830e-01 9.999725e-01 9.994396e-01
##  [26] 5.328637e-02 9.909700e-01 9.494277e-02 1.567069e-07 3.981355e-03
##  [31] 3.856080e-04 1.813525e-06 9.998344e-01 2.209306e-07 2.954528e-03
##  [36] 5.070327e-05 9.993336e-01 1.000000e+00 5.158232e-09 3.876064e-09
##  [41] 9.999201e-01 3.806671e-08 8.243606e-01 9.999975e-01 9.679253e-01
```

```
##  [46] 1.000000e+00 6.151135e-01 9.999437e-01 9.999996e-01 1.620713e-05
##  [51] 1.000000e+00 2.747063e-05 4.103830e-11 3.120788e-05 1.000000e+00
##  [56] 9.967275e-01 1.000000e+00 9.999999e-01 3.994717e-08 3.486068e-04
##  [61] 9.998150e-01 3.655903e-02 1.513594e-05 9.147647e-01 9.999999e-01
##  [66] 9.999935e-01 9.999977e-01 9.303714e-01 9.998878e-01 1.000000e+00
##  [71] 9.999889e-01 1.623858e-06 2.147278e-04 9.999999e-01 9.981876e-01
##  [76] 9.999973e-01 1.000000e+00 1.000000e+00 9.999999e-01 1.718324e-03
##  [81] 1.000000e+00 3.592877e-07 8.398993e-06 4.790128e-06 9.999860e-01
##  [86] 1.632298e-07 1.172576e-06 5.633606e-09 3.947319e-06 9.279134e-03
##  [91] 1.000000e+00 1.803884e-07 9.999308e-01 4.720263e-07 9.999986e-01
##  [96] 8.996179e-07 9.994213e-01 1.000000e+00 1.380322e-05 4.796470e-03
## [101] 9.999983e-01 1.000000e+00 1.000000e+00 9.999997e-01 1.296775e-06
## [106] 9.999999e-01 5.072680e-07 1.202301e-06 9.494792e-04 9.999647e-01
## [111] 9.999998e-01 9.999990e-01 1.000000e+00 9.985365e-01 3.083616e-04
## [116] 1.156950e-06 1.000000e+00 2.593751e-01 2.925558e-08 2.107466e-06
## [121] 9.999999e-01 1.464992e-04 1.869833e-09 9.999805e-01 5.427936e-02
## [126] 1.134842e-06 1.000000e+00 1.046174e-02 9.999976e-01 4.137581e-04
## [131] 9.999838e-01 4.704954e-04 9.999737e-01 9.997377e-01 1.000000e+00
## [136] 1.000000e+00 7.970898e-03 8.300408e-05 1.972532e-03 9.999992e-01
## [141] 1.000000e+00 1.052351e-04 4.567945e-03 9.999559e-01 1.564167e-04
## [146] 1.000000e+00 9.988810e-01 2.750594e-02 7.470651e-01 1.000000e+00
## [151] 6.516132e-10 9.224144e-10 9.999998e-01 9.999973e-01 2.436049e-07
## [156] 3.423623e-03 9.999980e-01 9.976740e-01 3.928747e-06 1.437425e-04
## [161] 1.000000e+00 1.783943e-03 1.000000e+00 6.677953e-08 9.777443e-01
## [166] 1.455892e-03 4.455770e-01 1.704991e-01 9.830385e-05 1.000000e+00
## [171] 8.721318e-07 1.000000e+00 1.000000e+00 2.043317e-05 1.405303e-01
## [176] 9.999959e-01 6.336607e-10 9.999987e-01 1.000658e-05 4.973547e-02
## [181] 3.078647e-08 2.882982e-07 6.025413e-06 1.000000e+00 1.448956e-02
## [186] 5.354844e-04 2.281256e-06 3.478447e-05 2.062303e-08 2.598415e-02
## [191] 1.074793e-06 9.999849e-01 1.393536e-06 9.999838e-01 2.311954e-02
## [196] 9.842965e-01 5.674371e-08 4.872614e-03 1.659406e-01 7.628267e-01
## [201] 2.441643e-01 9.999923e-01 1.000000e+00 2.063161e-02 9.999470e-01
## [206] 9.999997e-01 1.000000e+00 1.741239e-03 6.337190e-01 5.771366e-05
## [211] 9.999967e-01 9.999964e-01 1.758441e-04 1.629845e-02 1.736101e-03
## [216] 9.999448e-01 9.999999e-01 3.497493e-05 1.000000e+00 1.856491e-02
## [221] 1.637105e-03 9.999992e-01 2.863481e-08 9.999992e-01 1.000000e+00
## [226] 1.467252e-04 9.954800e-01 1.000000e+00 1.000000e+00 9.996056e-01
## [231] 1.000000e+00 9.999742e-01 2.708882e-06 8.792525e-08 9.997138e-01
## [236] 5.945875e-02 7.769166e-01 1.000000e+00 9.998660e-01 1.054787e-06
## [241] 7.539982e-11 1.182737e-05 2.284552e-06 9.999999e-01 7.598611e-06
## [246] 1.000000e+00 9.999876e-01 3.087539e-05 6.169957e-01 4.658843e-01
## [251] 9.999999e-01 9.953593e-01 1.598147e-01 9.998746e-01 9.891653e-01
## [256] 7.522358e-06 2.199683e-08 1.000000e+00 9.999936e-01 9.405450e-01
## [261] 9.999990e-01 1.000000e+00 1.000000e+00 2.470501e-01 5.249262e-04
## [266] 9.871624e-01 1.000000e+00 9.999783e-01 9.999944e-01 8.921258e-07
## [271] 7.700681e-08 7.380295e-06 4.698145e-05 2.590483e-05 9.998948e-01
## [276] 1.933857e-01 9.996233e-01 9.999991e-01 9.908209e-01 1.498433e-05
## [281] 4.395691e-06 9.999983e-01 2.005371e-04 2.161285e-03 5.798747e-09
## [286] 3.762045e-06 9.493398e-06 1.510653e-07 2.175792e-06 1.846434e-06
## [291] 1.000000e+00 9.999997e-01 3.671664e-07 7.890127e-06 9.995143e-05
## [296] 9.999744e-01 1.035364e-06 9.920998e-01 1.397753e-04 4.944288e-04
```

```
# 3) Use caret::confusionMatrix to construct the confusion matrix for your Naive Bayes Classifier Model
cm_nb = caret::confusionMatrix(y_pred_nb, as.factor(y_test$gender))
```

```r
# 4) Use K-Nearest Neighbor for predicting whether the customer was male or female?

# Feature Scaling
x_train_scale <- scale(x_train)
x_test_scale <- scale(x_test)

# Fitting K-NN to training set
# get best k
best_accuracy <- 0
best_k <- 0
for (i in 1:50) {
    knn_model <- caret::knn3(x = x_train_scale, y = factor(y_train$gender), k = i, prob = TRUE)
    y_pred_knn <- predict(knn_model, newdata = x_test_scale, type = "class")
    # print accuracy and k
    accuracy <- mean(y_pred_knn == y_test$gender)
    if (accuracy > best_accuracy) {
        best_accuracy <- accuracy
        best_k <- i
    }
    # print(paste("Accuracy is", mean(y_pred_knn == y_test$gender), "for k =", i))
}
print(paste("Best accuracy is", best_accuracy, "for k =", best_k))
```

```
## [1] "Best accuracy is 0.95 for k = 6"
```

```r
knn_model <- caret::knn3(x = x_train_scale, y = factor(y_train$gender), k = best_k, prob = TRUE)
knn_model
```

```
## 6-nearest neighbor model
## Training set outcome distribution:
##
##   F   M
## 342 358
```

```r
y_pred_knn <- predict(knn_model, newdata = x_test_scale, type = "class")
y_pred_knn
```

```
##   [1] M F F M M M M M F F M F F M F M F M F F M F M M M F M F F F F F M F F F M
##  [38] M F F M F F M F M M M M F M F F F M M M M F F M M F M M M M M M M M F F M
##  [75] M M M M M F M F F F M F F F F F M F M F M F M M F F M M M M F M F F F M M
## [112] M M M F F M M F F M F F M F F M F M F M F M M M M F F F M M F F M F M M F
## [149] M M F F M M F F M M F F M F M F M F F F F M F M M F F M F M F F F F F M F
## [186] F F F F F F M F M M M F F F M F M M F M M M F M F M F F M F M F M F F M M
## [223] F M M F M M M M M M F F M F M M M F F F F M F M M F M F M M F M M F F M M
## [260] M M M M F F M M M M F F F F F M F M M M F F M F F F F F F F F M M F F F M
## [297] F F F F
## Levels: F M
```

```r
# getting probability of M for ROC
y_pred_knn_probM <- predict(knn_model, newdata = x_test_scale)[,2]
y_pred_knn_probM
```

```
##   [1] 1.0000000 0.0000000 0.0000000 1.0000000 1.0000000 1.0000000 1.0000000
##   [8] 0.6666667 0.0000000 0.0000000 1.0000000 0.0000000 0.3333333 1.0000000
##  [15] 0.0000000 1.0000000 0.5000000 1.0000000 0.0000000 0.0000000 0.8333333
```

```
##  [22] 0.1666667 1.0000000 1.0000000 1.0000000 0.3333333 0.8333333 0.1666667
##  [29] 0.0000000 0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.5000000
##  [36] 0.1666667 0.8333333 1.0000000 0.0000000 0.0000000 1.0000000 0.0000000
##  [43] 0.5000000 1.0000000 0.5000000 1.0000000 0.6666667 1.0000000 1.0000000
##  [50] 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 1.0000000 0.6666667
##  [57] 1.0000000 1.0000000 0.0000000 0.0000000 1.0000000 0.6666667 0.3333333
##  [64] 0.6666667 1.0000000 1.0000000 1.0000000 0.8333333 0.8333333 1.0000000
##  [71] 1.0000000 0.0000000 0.0000000 1.0000000 0.8333333 1.0000000 1.0000000
##  [78] 1.0000000 1.0000000 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000
##  [85] 1.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 1.0000000
##  [92] 0.0000000 0.8333333 0.0000000 1.0000000 0.0000000 0.6666667 1.0000000
##  [99] 0.0000000 0.3333333 0.6666667 1.0000000 1.0000000 1.0000000 0.0000000
## [106] 1.0000000 0.0000000 0.0000000 0.1666667 0.8333333 1.0000000 1.0000000
## [113] 1.0000000 1.0000000 0.0000000 0.0000000 1.0000000 0.5000000 0.0000000
## [120] 0.0000000 1.0000000 0.0000000 0.0000000 1.0000000 0.3333333 0.0000000
## [127] 1.0000000 0.0000000 1.0000000 0.0000000 1.0000000 0.0000000 0.8333333
## [134] 0.8333333 1.0000000 1.0000000 0.3333333 0.0000000 0.0000000 1.0000000
## [141] 1.0000000 0.0000000 0.5000000 1.0000000 0.0000000 1.0000000 1.0000000
## [148] 0.1666667 0.5000000 1.0000000 0.0000000 0.0000000 0.8333333 1.0000000
## [155] 0.3333333 0.1666667 0.8333333 0.6666667 0.0000000 0.3333333 1.0000000
## [162] 0.0000000 1.0000000 0.0000000 0.6666667 0.1666667 0.5000000 0.3333333
## [169] 0.0000000 1.0000000 0.0000000 1.0000000 1.0000000 0.0000000 0.5000000
## [176] 1.0000000 0.0000000 1.0000000 0.0000000 0.3333333 0.0000000 0.0000000
## [183] 0.0000000 1.0000000 0.3333333 0.3333333 0.0000000 0.0000000 0.0000000
## [190] 0.1666667 0.0000000 0.8333333 0.0000000 0.6666667 0.5000000 0.6666667
## [197] 0.0000000 0.0000000 0.3333333 0.6666667 0.1666667 1.0000000 1.0000000
## [204] 0.0000000 0.6666667 1.0000000 1.0000000 0.1666667 0.5000000 0.0000000
## [211] 1.0000000 1.0000000 0.1666667 0.1666667 0.5000000 1.0000000 1.0000000
## [218] 0.1666667 1.0000000 0.1666667 0.1666667 0.8333333 0.0000000 1.0000000
## [225] 1.0000000 0.0000000 1.0000000 1.0000000 1.0000000 0.8333333 1.0000000
## [232] 0.8333333 0.0000000 0.0000000 1.0000000 0.3333333 0.6666667 1.0000000
## [239] 0.8333333 0.0000000 0.0000000 0.0000000 0.0000000 1.0000000 0.0000000
## [246] 1.0000000 0.8333333 0.0000000 0.6666667 0.3333333 1.0000000 0.8333333
## [253] 0.5000000 1.0000000 0.8333333 0.0000000 0.0000000 1.0000000 1.0000000
## [260] 0.6666667 0.8333333 1.0000000 1.0000000 0.3333333 0.1666667 1.0000000
## [267] 1.0000000 1.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## [274] 0.0000000 1.0000000 0.1666667 0.6666667 0.8333333 0.5000000 0.0000000
## [281] 0.0000000 1.0000000 0.0000000 0.3333333 0.1666667 0.0000000 0.0000000
## [288] 0.0000000 0.0000000 0.0000000 1.0000000 1.0000000 0.1666667 0.0000000
## [295] 0.0000000 1.0000000 0.0000000 0.3333333 0.0000000 0.1666667
```

```
# 5) Use caret::confusionMatrix to construct the confusion matrix for your K-Nearest Neighbor
cm_knn = caret::confusionMatrix(y_pred_knn, as.factor(y_test$gender))
cm_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   F   M
##          F 145   7
##          M   8 140
##
##                Accuracy : 0.95
##                  95% CI : (0.9189, 0.9717)
##     No Information Rate : 0.51
```

```
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9477
##             Specificity : 0.9524
##          Pos Pred Value : 0.9539
##          Neg Pred Value : 0.9459
##              Prevalence : 0.5100
##          Detection Rate : 0.4833
##    Detection Prevalence : 0.5067
##       Balanced Accuracy : 0.9500
##
##        'Positive' Class : F
##
```

```
# 6) Compare your two models using sensitivity, specificity.
# get sensitivity, specificity for NB
sensitivity_nb <- cm_nb$byClass[1]
print(paste("Sensitivity for Naive Bayes is", sensitivity_nb))
```

```
## [1] "Sensitivity for Naive Bayes is 0.947712418300654"
```

```
specificity_nb <- cm_nb$byClass[2]
print(paste("Specificity for Naive Bayes is", specificity_nb))
```

```
## [1] "Specificity for Naive Bayes is 0.952380952380952"
```

```
# get sensitivity, specificity for KNN
sensitivity_knn <- cm_knn$byClass[1]
print(paste("Sensitivity for K-Nearest-Neighbors is", sensitivity_knn))
```

```
## [1] "Sensitivity for K-Nearest-Neighbors is 0.947712418300654"
```

```
specificity_knn <- cm_knn$byClass[2]
print(paste("Specificity for K-Nearest-Neighbors is", specificity_knn))
```

```
## [1] "Specificity for K-Nearest-Neighbors is 0.952380952380952"
```

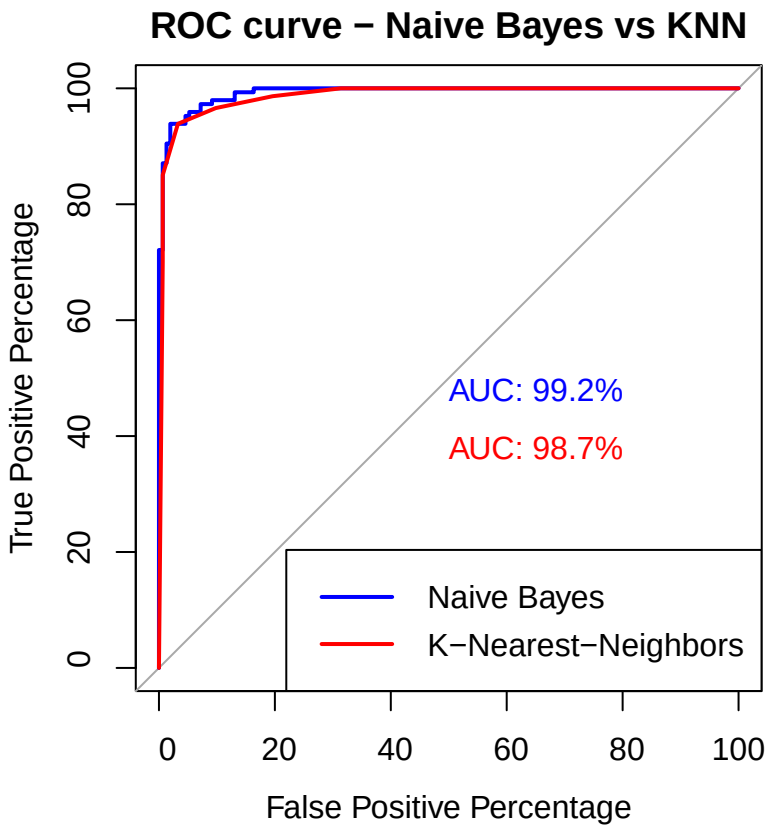
```
# 7) Compare the ROC charts of your two models.
par(pty="s")

roc_nb <- pROC::roc(y_test$gender, y_pred_nb_probM, plot=TRUE, legacy.axes = TRUE, xlab="False Positive
```

```
## Setting levels: control = F, case = M
## Setting direction: controls < cases
```

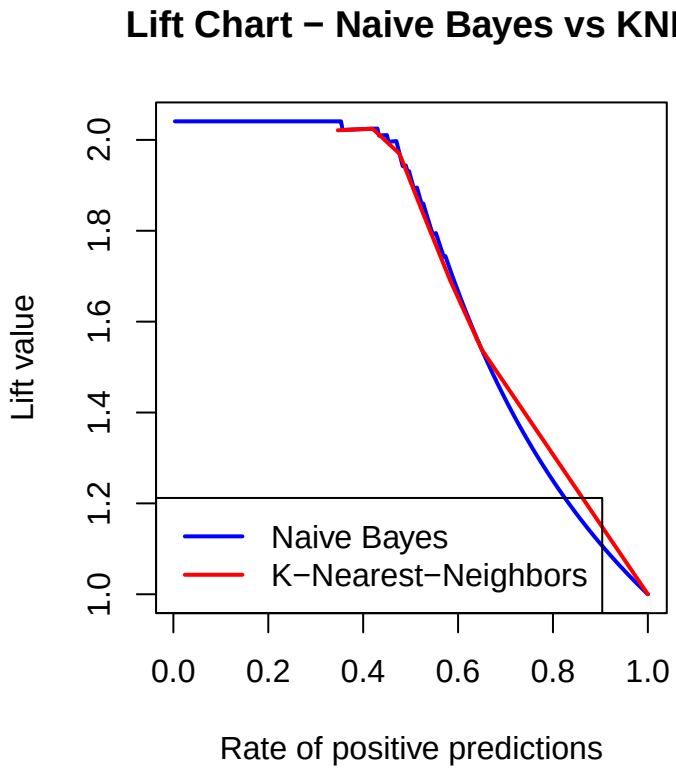
```
plot.roc(y_test$gender, y_pred_knn_probM, percent=TRUE, col="red", lwd=2, print.auc=TRUE, add=TRUE, pri
```

```
## Setting levels: control = F, case = M
## Setting direction: controls < cases
```

```
legend("bottomright", legend = c("Naive Bayes", "K-Nearest-Neighbors"), col = c("blue", "red"), lwd = 2)
```




```
# 8) Compare the lift charts of your two models.
# lift chart for Naive Bayes
pred_nb <- prediction(y_pred_nb_probM, y_test$gender)
perf_nb <- performance(pred_nb, "lift", "rpp")
plot(perf_nb, main="Lift Chart - Naive Bayes vs KNN", col="blue", lwd=2)

# lift chart for KNN
pred_knn <- prediction(y_pred_knn_probM, y_test$gender)
perf_knn <- performance(pred_knn, "lift", "rpp")
plot(perf_knn, col="red", lwd=2, add=TRUE)

legend("bottomleft", legend = c("Naive Bayes", "K-Nearest-Neighbors"), col = c("blue", "red"), lwd = 2)
```


Lift Chart − Naive Bayes vs KNN
Lift value
1.0
1.2
1.4
1.6
1.8
2.0
0.0
0.2
0.4
0.6
0.8
1.0
Rate of positive predictions
Naive Bayes
K−Nearest−Neighbors