CPTR330 – Homework 2

Pierre Visconti

April 10, 2023

Naive Bayes Algorithm

Naive Bayes is a supervised probabilistic classification algorithm based on Bayes' theorem for calculating conditional probability. It assumes that all of the features in the data are independent from one another and each are equally important.

Some strengths of Naive Bayes is that it performs as well as some of the best classifier algorithms even while violating the assumption of independence and it is robust to noisy and missing data. Naive Bayes though is not ideal for datasets with a lot of numeric features as they will all have to be converted to factors in some way, also the assumptions it makes are rarely ever true in the real world.

Step 1 - Collect Data

The data was extracted by Barry Becker from the 1994 census bureau database. The goal is the dataset is to predict whether someone's income is greater or less than 50k. The dataset includes a number of variables like each person's education level, their work occupation, their sex, the hours they work per week, etc.

Step 2 - Exploring And Preparing The Data

The dataset has 6 numerical variables, and 9 categorical variables. The categorical variables can simply be converted to factors, but the numerical variables require further transformation to be usable by Naive Bayes. There are many missing values in the dataset but thankfully Naive Bayes is very robust in this regard. The dataset also does not include column names so that needs to be added in as well.

Importing data and naming columns

```
# import data
adult = read.csv("adult.data")
# changing column names to the actual names as given on the UCI website
colnames(adult) = c('age', 'workclass', 'fnlwgt', 'education', 'education_num',
                    'married', 'occupation', 'relationship', 'race', 'sex',
                    'capital_gain', 'capital_loss', 'hrs_week', 'native_country'
                    , 'income')
str(adult)
```

```
## 'data.frame':    32560 obs. of  15 variables:
##  $ age           : int  50 38 53 28 37 49 52 31 42 37 ...
##  $ workclass     : chr  " Self-emp-not-inc" " Private" " Private" " Private" ...
##  $ fnlwgt        : int  83311 215646 234721 338409 284582 160187 209642 45781 159449 280464 ...
##  $ education     : chr  " Bachelors" " HS-grad" " 11th" " Bachelors" ...
##  $ education_num : int  13 9 7 13 14 5 9 14 13 10 ...
##  $ married       : chr  " Married-civ-spouse" " Divorced" " Married-civ-spouse" " Married-civ-spouse
##  $ occupation    : chr  " Exec-managerial" " Handlers-cleaners" " Handlers-cleaners" " Prof-specialty
##  $ relationship  : chr  " Husband" " Not-in-family" " Husband" " Wife" ...
```

```
##  $ race          : chr  " White" " White" " Black" " Black" ...
##  $ sex           : chr  " Male" " Male" " Male" " Female" ...
##  $ capital_gain  : int  0 0 0 0 0 0 0 14084 5178 0 ...
##  $ capital_loss  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hrs_week      : int  13 40 40 40 40 16 45 50 40 80 ...
##  $ native_country: chr  " United-States" " United-States" " United-States" " Cuba" ...
##  $ income        : chr  " <=50K" " <=50K" " <=50K" " <=50K" ...
```

We need to convert each feature to a factor.

```
# converting categorical variables
adult$workclass = as.factor(adult$workclass)
adult$education = as.factor(adult$education)
adult$married = as.factor(adult$married)
adult$occupation = as.factor(adult$occupation)
adult$relationship = as.factor(adult$relationship)
adult$race = as.factor(adult$race)
adult$sex = as.factor(adult$sex)
adult$native_country = as.factor(adult$native_country)
adult$income = as.factor(adult$income)
```

```
str(adult)
```

```
## 'data.frame':    32560 obs. of  15 variables:
##  $ age           : int  50 38 53 28 37 49 52 31 42 37 ...
##  $ workclass     : Factor w/ 9 levels " ?"," Federal-gov",..: 7 5 5 5 5 5 7 5 5 5 ...
##  $ fnlwgt        : int  83311 215646 234721 338409 284582 160187 209642 45781 159449 280464 ...
##  $ education     : Factor w/ 16 levels " 10th"," 11th",..: 10 12 2 10 13 7 12 13 10 16 ...
##  $ education_num : int  13 9 7 13 14 5 9 14 13 10 ...
##  $ married       : Factor w/ 7 levels " Divorced"," Married-AF-spouse",..: 3 1 3 3 3 4 3 5 3 3 ...
##  $ occupation    : Factor w/ 15 levels " ?"," Adm-clerical",..: 5 7 7 11 5 9 5 11 5 5 ...
##  $ relationship  : Factor w/ 6 levels " Husband"," Not-in-family",..: 1 2 1 6 6 2 1 2 1 1 ...
##  $ race          : Factor w/ 5 levels " Amer-Indian-Eskimo",..: 5 5 3 3 5 3 5 5 5 3 ...
##  $ sex           : Factor w/ 2 levels " Female"," Male": 2 2 2 1 1 1 2 1 2 2 ...
##  $ capital_gain  : int  0 0 0 0 0 0 0 14084 5178 0 ...
##  $ capital_loss  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hrs_week      : int  13 40 40 40 40 16 45 50 40 80 ...
##  $ native_country: Factor w/ 42 levels " ?"," Cambodia",..: 40 40 40 6 40 24 40 40 40 40 ...
##  $ income        : Factor w/ 2 levels " <=50K"," >50K": 1 1 1 1 1 1 2 2 2 2 ...
```

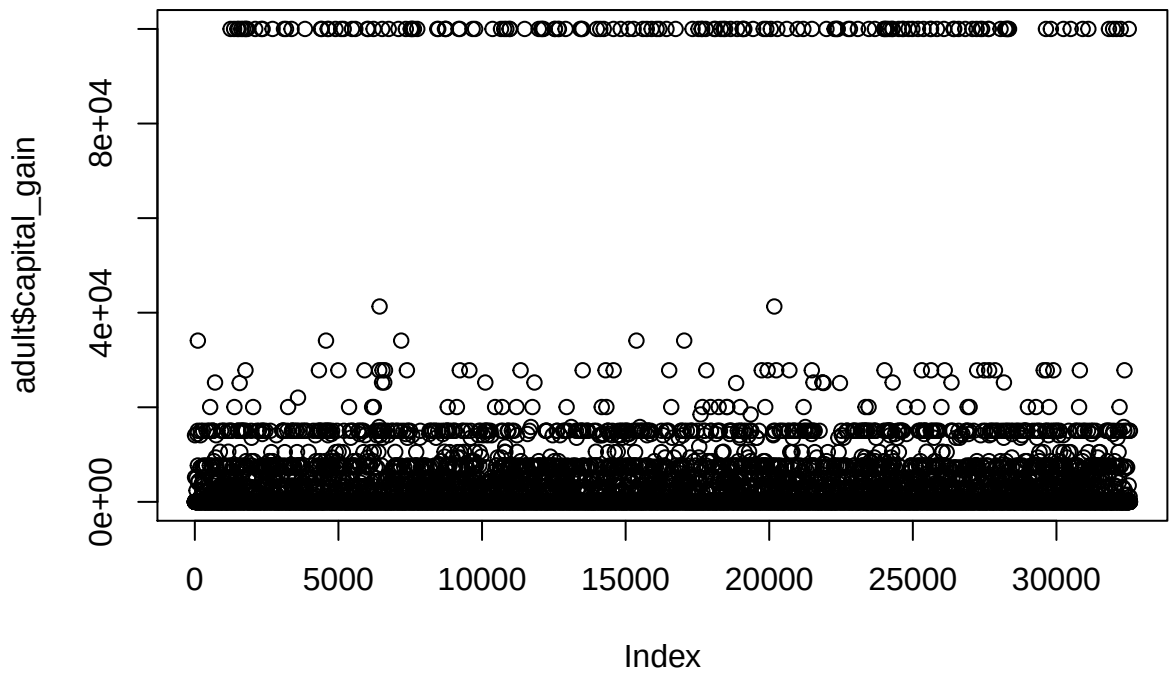
All the categorical variables have been converted to factors, the numerical variables now need to be converted.

```
summary(adult[c('age', 'fnlwgt', 'education_num', 'capital_gain', 'capital_loss'
                , 'hrs_week')])
```

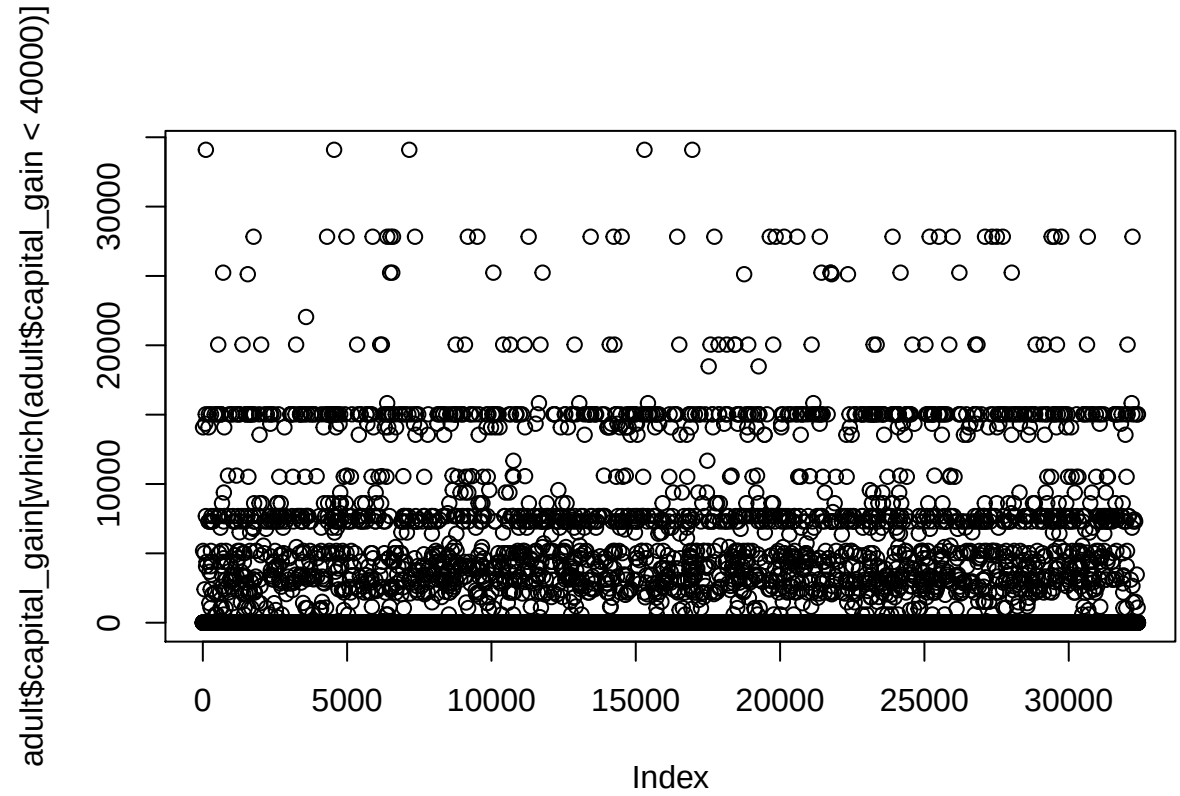
```
##       age            fnlwgt        education_num    capital_gain
##  Min.   :17.00   Min.   :  12285   Min.   : 1.00   Min.   :    0
##  1st Qu.:28.00   1st Qu.: 117832   1st Qu.: 9.00   1st Qu.:    0
##  Median :37.00   Median : 178363   Median :10.00   Median :    0
##  Mean   :38.58   Mean   : 189782   Mean   :10.08   Mean   : 1078
##  3rd Qu.:48.00   3rd Qu.: 237054   3rd Qu.:12.00   3rd Qu.:    0
##  Max.   :90.00   Max.   :1484705   Max.   :16.00   Max.   :99999
##   capital_loss       hrs_week
##  Min.   :   0.00   Min.   : 1.00
##  1st Qu.:   0.00   1st Qu.:40.00
##  Median :   0.00   Median :40.00
##  Mean   :  87.31   Mean   :40.44
```

```
##  3rd Qu.:   0.00   3rd Qu.:45.00
##  Max.   :4356.00   Max.   :99.00
```

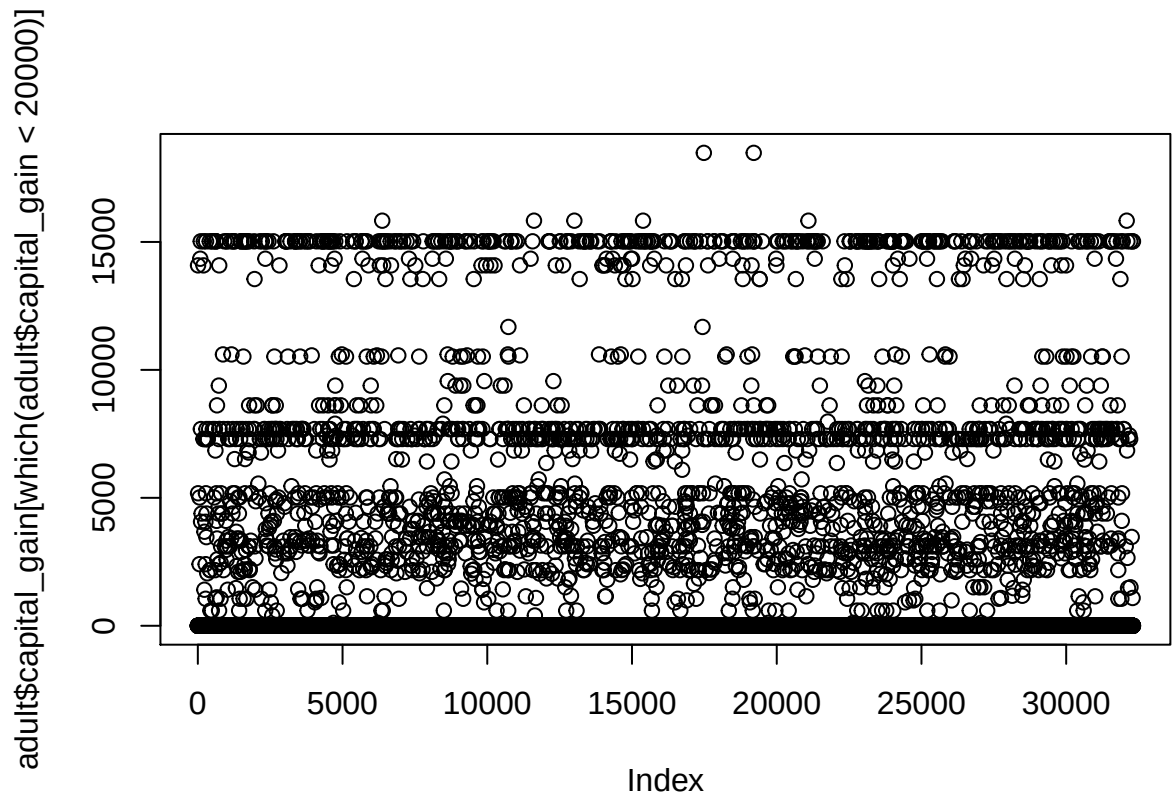
```
plot(adult$capital_gain)
```



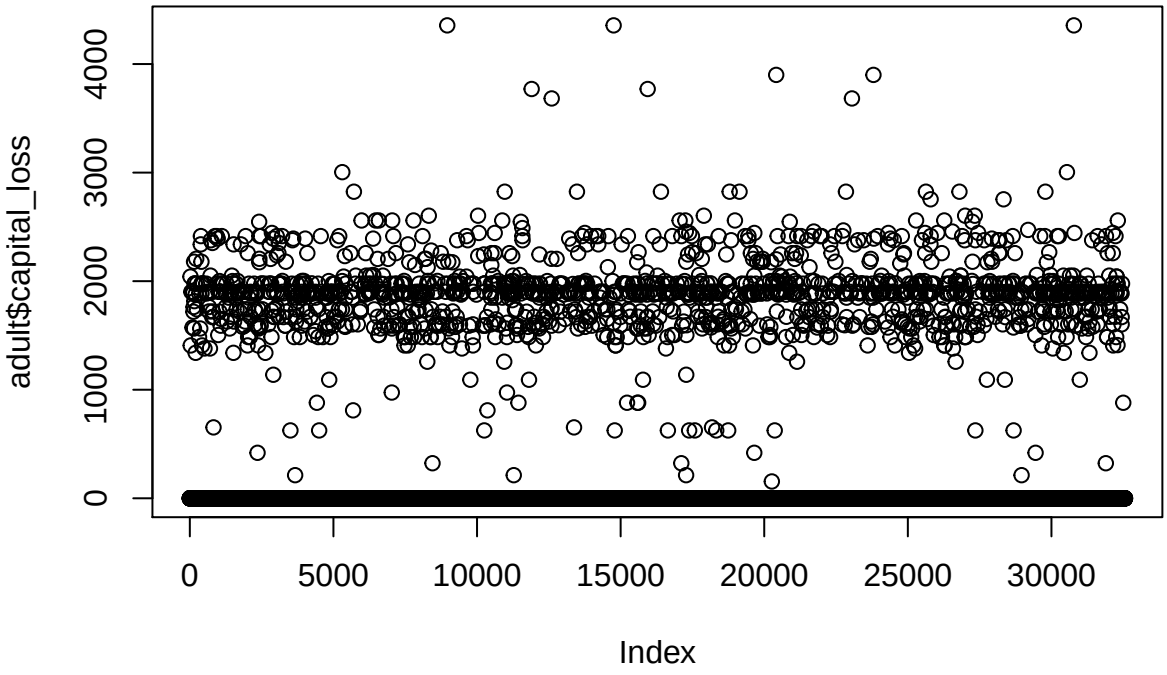

```
plot(adult$capital_gain[which(adult$capital_gain<40000)])
```



```r
plot(adult$capital_gain[which(adult$capital_gain<20000)])
```




```r
# appears to be divisible into 6 groups
plot(adult$capital_loss)
```




```r
# and 4 groups in capital_loss
```

Binning numerical variables

```r
# binning age by significant age levels. Note: 18 was chosen as that is age of
# adult, and 67 was chosen as that is the age of retirement in the US to
# receive full benefits.
adult = adult %>% mutate(age = cut(age, breaks = c(0,18,30,40,50,67,90)))
# binning fnlwgt into 10 bins
adult = adult %>% mutate(fnlwgt = ntile(fnlwgt, n=10))
adult$fnlwgt = as.factor(adult$fnlwgt)
# binning education_number by diploma level
adult = adult %>% mutate(education_num = cut(education_num,
        breaks = c(0,8,12,16)))
# binning capital_gain into the 6 groups observed above
adult = adult %>% mutate(capital_gain = cut(capital_gain,
        breaks = c(-1, 2000,5500,12000,16000,45000,max(adult$capital_gain))))
        #-1 used to include entries with value 0
# binning capital_loss into the 4 grouos observed above
adult = adult %>% mutate(capital_loss = cut(capital_loss,
        breaks = c(-1,100,1300,3000,max(adult$capital_loss))))
        #-1 used to include entries with value 0
# binning hrs_week by typical work week hours for not working, part time,
# full time, and over typical full-time.
adult = adult %>% mutate(hrs_week = cut(hrs_week,
        breaks = c(-1,30,40,50,60,max(adult$hrs_week))))
str(adult)
```

```
## 'data.frame':    32560 obs. of  15 variables:
##  $ age           : Factor w/ 6 levels "(0,18]","(18,30]",..: 4 3 5 2 3 4 5 3 4 3 ...
##  $ workclass     : Factor w/ 9 levels " ?"," Federal-gov",..: 7 5 5 5 5 5 7 5 5 5 ...
##  $ fnlwgt        : Factor w/ 10 levels "1","2","3","4",..: 2 7 8 10 9 5 7 1 5 9 ...
##  $ education     : Factor w/ 16 levels " 10th"," 11th",..: 10 12 2 10 13 7 12 13 10 16 ...
##  $ education_num : Factor w/ 3 levels "(0,8]","(8,12]",..: 3 2 1 3 3 1 2 3 3 2 ...
##  $ married       : Factor w/ 7 levels " Divorced"," Married-AF-spouse",..: 3 1 3 3 3 4 3 5 3 3 ...
##  $ occupation    : Factor w/ 15 levels " ?"," Adm-clerical",..: 5 7 7 11 5 9 5 11 5 5 ...
##  $ relationship  : Factor w/ 6 levels " Husband"," Not-in-family",..: 1 2 1 6 6 2 1 2 1 1 ...
##  $ race          : Factor w/ 5 levels " Amer-Indian-Eskimo",..: 5 5 3 3 5 3 5 5 5 3 ...
##  $ sex           : Factor w/ 2 levels " Female"," Male": 2 2 2 1 1 1 2 1 2 2 ...
##  $ capital_gain  : Factor w/ 6 levels "(-1,2e+03]","(2e+03,5.5e+03]",..: 1 1 1 1 1 1 1 4 2 1 ...
##  $ capital_loss  : Factor w/ 4 levels "(-1,100]","(100,1.3e+03]",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ hrs_week      : Factor w/ 5 levels "(-1,30]","(30,40]",..: 1 2 2 2 2 1 3 3 2 5 ...
##  $ native_country: Factor w/ 42 levels " ?"," Cambodia",..: 40 40 40 6 40 24 40 40 40 40 ...
##  $ income        : Factor w/ 2 levels " <=50K"," >50K": 1 1 1 1 1 1 2 2 2 2 ...
```

create training and testing datasets

```r
# setting seed so that results do not change when knitting to pdf
set.seed(1)
# create training and test data by randomly sampling
train.size = round(nrow(adult)*0.8) # 80% of the dataset used for training
train.ind = sample(1:nrow(adult), train.size)
adult_train = adult[train.ind,-15]
adult_test = adult[-train.ind,-15]
# create labels for training and test data
adult_train_labels = adult[train.ind, 15]
adult_test_labels = adult[-train.ind, 15]
```

The data is now fully prepared for training.

Step 3 - Training A Model On The Data

Training the model involves feeding the dataset of the features and the corresponding labels to the model. Using these datasets the model calculates a probability for every single feature which it can then use to make predictions by calculating the probability of the input belonging to a class. Since this only requires basic mathematical operations the process is extremely quick compared to other algorithms.

```
income_classifier = naiveBayes(adult_train[,c(2,4,8,14)], adult_train_labels)
```

Step 4 - Evaluating Model Performance

```
income_test_pred = predict(income_classifier, adult_test[,c(2,4,8,14)])
table(income_test_pred, adult_test_labels)
```

```
##                 adult_test_labels
## income_test_pred  <=50K  >50K
##            <=50K   4600   860
##            >50K     313   739
```

```
CrossTable(income_test_pred, adult_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  6512
##
##
##              | actual
##    predicted |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
##        <=50K |      4600 |       860 |      5460 |
##              |     0.706 |     0.132 |           |
## -------------|-----------|-----------|-----------|
##         >50K |       313 |       739 |      1052 |
##              |     0.048 |     0.113 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      4913 |      1599 |      6512 |
## -------------|-----------|-----------|-----------|
##
##
```

The model predicted someone's income being greater than 50k, 4600 times out of the 5460 people with income greater than 50k in the test set. It predicted someone's income being less than 50k, 739 times out of the total 1052 people with income less than 50k in the dataset. The model achieved an accuracy of 0.82 on the test dataset.

Step 5 - Improving Model Performance

There are several options available to improve the model. The first is that we can implement a laplace smoothing to overcome the issue of 0 probability cases. We can also try to include more features in the model and see if we get better results. Lastly we can play around with the way that the bins for the numerical values were chosen.

```
### List of different model performances:
# original: 81.98%
# original with laplace=1: 82.04%
# with capital_gain added: 84%
# with capital_gain and capital_loss added: 84.3%
# with occupation:  worse performance
# with fnlwgt: no improvement
# with sex: worse performance
# with race: no improvement
# with hrs_week: worse performance

income_classifier2 = naiveBayes(adult_train[,c(2,4,8,9,11,12,14)], adult_train_labels, laplace = 1)
income_test_pred2 = predict(income_classifier2, adult_test[,c(2,4,8,9,11,12,14)])
table(income_test_pred2, adult_test_labels)
```

```
##                  adult_test_labels
## income_test_pred2  <=50K  >50K
##             <=50K   4600   707
##             >50K     313   892
```

```
CrossTable(income_test_pred2, adult_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  6512
##
##
##              | actual
##    predicted |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
##        <=50K |      4600 |       707 |      5307 |
##              |     0.706 |     0.109 |           |
## -------------|-----------|-----------|-----------|
##         >50K |       313 |       892 |      1205 |
##              |     0.048 |     0.137 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      4913 |      1599 |      6512 |
## -------------|-----------|-----------|-----------|
##
##
```

Adding a laplace smoothing to the model helped improve results slightly to 0.8204 from 0.8198. Adding capital_gain to the model further improved results up to 0.84 and with capital_loss included as well, prediction accuracy on the test dataset was 0.843. I then tried additional variables one at a time, like sex, hrs_week, and occupation, but they all either made the model worse, or had no effect.

The final model added a laplace smoothing, capital_gain, capital_loss and managed to achieve an accuracy of 0.843 compared to the original model's 0.82.

Autograding

```
.AutograderMyTotalScore()
```

```
##
## Step 1:      0/1
## Step 2:      0/1
## Step 3:      0/1
## Step 4:      0/1
## Step 5:      0/1
## Total Score: 0/5
```