MATH 3190 Homework 3

Focus: Notes 6

Due February 24, 2024

Your homework should be completed in R Markdown or Quarto and Knitted to an html or pdf document. You will "turn in" this homework by uploading to your GitHub Math_3190_Assignment repository in the Homework directory.

Problem 1 (19 points)

Part a (16 points)

Suppose we are attempting to predict a person's ability to run a marathon in under 4 hours (coded as a 1) based on a number of factors: age, sex, BMI, and blood pressure. Below is the confusion matrix in this situation:

		Actual	
		1	0
Predicted	1	58	102
	0	37	217

Find each of the following. Use proper formatting in R Markdown when you type your answers. You can put equations between dollar signs (`$$`) and you can use the `\frac{}{}` (for a small fraction) or `\dfrac{}{}` (for a larger one) commands to nicely type fractions.

- The prevalence of those that can run a mile (?marathon?) under 4 hours.
- The overall accuracy of these predictions.
- The sensitivity (recall).
- The specificity.
- The positive predictive value (precision).
- The negative predictive value.
- The balanced accuracy.
- Cohen's Kappa (κ). Check out this link on Wikipedia and scroll down to the section entitled **Binary classification confusion matrix**.

• Prevalence (frequency) of those capable of 4-hour marathon: $58 + 37 = 95$

and prevalence as relative frequency : $\frac{95}{414} = 0.22947$ or 22.95%

• Overall accuracy : $\dfrac{58 + 217}{58 + 102 + 37 + 217} = \dfrac{275}{414} = 0.66425$ or 66.53%

• Sensitivity/recall : $\dfrac{58}{95} = 0.61053$ or 61.05%

• Specificity : $\dfrac{217}{319} = 0.68025$ or 68.03%

- Precision (PPV) : $\frac{58}{160} = 0.3625$ or 36.45%
- Negative Predictive Value : $\frac{217}{254} = 0.85433$ or 85.43%
- Balanced Accuracy : $\frac{0.61053 + 0.68025}{2} = 0.64539$ or 64.54%
- Cohen's Kappa (joint-probability of agreement): $\kappa = \frac{2 \times (TP \times TN - FN \times FP)}{(TP + FP) \times (FP + TN) + (TP + FN) \times (FN + TN)}$

$= \frac{2(58 \times 217 - 37 \times 102)}{(58 + 102) \times (102 + 217) + (58 + 37) \times (37 + 217)} = \frac{17624}{75170} = 0.23446$

Part b (3 points)

Read more of the Wikipedia article on Cohen's Kappa, especially the **Interpreting magnitude** and the **Limitations** part. I cannot really verify that you did this, so this is on your honor.

- P-value for /*kappa* is rarely reported because low values can be significantly different from zero but of insufficient magnitude
- Confidence intervalles may be constructed for a theoretical infinite number of items checked
- Prevalence and bias influence magnitude of κ: higher when codes ('ratings') are equiprobable but also higher when codes are asymmetrically distributed; higher when number of codes increases
- Thus guidelines of magnitude interpretation vary and are unsubstantiated
- May be informative to instead report quantity and allocation disagreement

Problem 2 (81 points)

The `adult` dataset (from the UC Irvine database) is one that is used to predict whether a person makes over $50K a year based on some other variables. The data came from the Census Bureau in 1994 and can be found in the Data folder in my Math3190_S24 GitHub repo. More info on the dataset can be found in the "adult.names" file.

Part a (5 points)

Read the data into **R** as a tibble, change the column names to be descriptive about what the variable in that column is, and change the one containing salary information to a factor. Read the "adult.names" file to see the column names.

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(caret)
```

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

```r
library(dplyr)
library(httr)
```

```
##
## Attaching package: 'httr'
##
## The following object is masked from 'package:caret':
##
##     progress
```

```r
response <- GET("https://raw.githubusercontent.com/rbrown53/Math3190_Sp24/main/Data/adult.data")

adult_data <- readr::read_csv(content(response, as = "text"),col_names = FALSE) |>
  rename(age = X1,
         work_class = X2,
         final_wght = X3,
         education = X4,
         edu_num = X5,
         marital = X6,
         occupation = X7,
         relationship = X8,
         race = X9,
         sex = X10,
         capital_gain = X11,
         capital_loss = X12,
         hrs_per_week = X13,
         country_origin = X14,
         salary = X15) |>
  mutate(salary = as_factor(salary),
         work_class = as_factor(work_class),
         education = as_factor(education),
         marital = as_factor(marital),
         occupation = as_factor(occupation),
         relationship = as_factor(relationship),
         race = as_factor(race),
         sex = as_factor(sex),
         country_origin = as_factor(country_origin))
```

```
## Rows: 48842 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (9): X2, X4, X6, X7, X8, X9, X10, X14, X15
```

```
## dbl (6): X1, X3, X5, X11, X12, X13
##
## 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.
```

```r
head(adult_data)
```

```
## # A tibble: 6 x 15
##     age work_class  final_wght education edu_num marital occupation relationship
##   <dbl> <fct>            <dbl> <fct>       <dbl> <fct>   <fct>      <fct>
## 1    39 State-gov        77516 Bachelors      13 Never-~ Adm-cleri~ Not-in-fami~
## 2    50 Self-emp-n~      83311 Bachelors      13 Marrie~ Exec-mana~ Husband
## 3    38 Private         215646 HS-grad         9 Divorc~ Handlers-~ Not-in-fami~
## 4    53 Private         234721 11th            7 Marrie~ Handlers-~ Husband
## 5    28 Private         338409 Bachelors      13 Marrie~ Prof-spec~ Wife
## 6    37 Private         284582 Masters        14 Marrie~ Exec-mana~ Wife
## # i 7 more variables: race <fct>, sex <fct>, capital_gain <dbl>,
## #   capital_loss <dbl>, hrs_per_week <dbl>, country_origin <fct>, salary <fct>
```

Part b (4 points)

Randomly split the dataset into a training and a testing group. Let's use 4/5 of it for training and 1/5 for testing. You can do this with any function you'd like. Please set a seed before you do this so the results are reproducible.

```r
y <- adult_data$salary
set.seed(2024)
train_index <- createDataPartition(y , times=1, p=0.8, list=FALSE)

train <- adult_data |>
  slice(train_index) |>
  select(-salary)
```

```
## Warning: Slicing with a 1-column matrix was deprecated in dplyr 1.1.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
y_train <- y[train_index]
y_train <- factor(y_train, levels = c(">50K", "<=50K"))

test <- adult_data |>
  slice(-train_index) |>
  select(-salary)

y_test <- y[-train_index]
y_test <- factor(y_test, levels = c(">50K", "<=50K"))
```

Part c (5 points)

Fit two models for predicting whether a person's salary is above $50K or not:

In the first, fit a logistic regression model using the `glm()` function with the `family` set to `"binomial"`. Use `age`, `education`, `race`, `sex`, and `hours_per_week` as the predictors.

```r
predictors <- train |>
  select(age, education, race, sex, hrs_per_week)

logit_model <- glm(y_train ~ . , family = 'binomial', data = predictors)

summary(logit_model)
```

```
##
## Call:
## glm(formula = y_train ~ ., family = "binomial", data = predictors)
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              3.288707   0.077871  42.233  < 2e-16 ***
## age                     -0.044377   0.001091 -40.667  < 2e-16 ***
## educationHS-grad         1.419098   0.038071  37.275  < 2e-16 ***
## education11th            2.420324   0.129751  18.654  < 2e-16 ***
## educationMasters        -0.380412   0.055548  -6.848 7.47e-12 ***
## education9th             2.730258   0.186775  14.618  < 2e-16 ***
## educationSome-college    0.979423   0.040509  24.178  < 2e-16 ***
## educationAssoc-acdm      0.591380   0.073687   8.026 1.01e-15 ***
## educationAssoc-voc       0.781969   0.066853  11.697  < 2e-16 ***
## education7th-8th         2.986783   0.153298  19.484  < 2e-16 ***
## educationDoctorate      -0.969026   0.115636  -8.380  < 2e-16 ***
## educationProf-school    -1.025619   0.098572 -10.405  < 2e-16 ***
## education5th-6th         2.911757   0.226584  12.851  < 2e-16 ***
## education10th            2.539021   0.134523  18.874  < 2e-16 ***
## education1st-4th         3.625956   0.419838   8.637  < 2e-16 ***
## educationPreschool       3.854151   1.014416   3.799 0.000145 ***
## education12th            1.937113   0.169960  11.397  < 2e-16 ***
## raceBlack                0.425511   0.056078   7.588 3.25e-14 ***
## raceAsian-Pac-Islander   0.180674   0.078022   2.316 0.020576 *
## raceAmer-Indian-Eskimo   0.628446   0.176283   3.565 0.000364 ***
## raceOther                0.390381   0.191480   2.039 0.041474 *
## sexFemale                1.122766   0.034450  32.591  < 2e-16 ***
## hrs_per_week            -0.035690   0.001186 -30.103  < 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: 43002  on 39073  degrees of freedom
## Residual deviance: 33366  on 39051  degrees of freedom
## AIC: 33412
##
## Number of Fisher Scoring iterations: 6
```

In the second, fit a k nearest neighbors model with $k = 7$ neighbors using the `knn3()` function in the `caret` package. Again, use `age`, `education`, `race`, `sex`, and `hours_per_week` as the predictors.

```r
knn_model <- knn3(y_train ~ . , k=7, data=predictors)
```

Part d (5 points)

With logistic regression, the most common cutoff value for the predicted probability for predicting a "success" is 0.5. Using 0.5 as this cutoff (above 0.5 should be labeled as ">50K"), obtain the class predictions and convert the variable to a factor. You can use the `predict()` function with `type = "response"` to obtain the predicted probabilities of being in the ">50K" group and then compare those probabilities to 0.5. Then use the `confusionMatrix()` function in the `caret` package to obtain the confusion matrix and many associated statistics. Print all of the output from that function.

```r
y_hat_logit <- predict(logit_model, type='response')
#probabilities predicted from training data

predicted_class <- factor(
  ifelse(y_hat_logit < 0.5, '>50K', '<=50K'), levels = c('>50K', '<=50K')
  )

cm_logit <- confusionMatrix(data = predicted_class, reference = y_train, positive = '>50K')

print(cm_logit$table)
```

```
##            Reference
## Prediction  >50K <=50K
##      >50K   3497  1900
##      <=50K  5853 27824
```

```r
print(cm_logit$overall)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.015816e-01   3.626331e-01   7.975922e-01   8.055262e-01   7.607104e-01
## AccuracyPValue  McnemarPValue
##   1.388858e-83   0.000000e+00
```

```r
print(cm_logit$byClass)
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.37401070           0.93607859           0.64795257
##       Neg Pred Value            Precision               Recall
##           0.82620186           0.64795257           0.37401070
##                   F1           Prevalence       Detection Rate
##           0.47426595           0.23928955           0.08949685
## Detection Prevalence    Balanced Accuracy
##           0.13812254           0.65504464
```

Part e (4 points)

Obtain the class predictions for your kNN model and output the results of the `confusionMatrix()` function for this. Note that it will take a few seconds to obtain the predictions for the kNN model.

```r
y_hat_knn <- predict(knn_model, predictors, type='class')


cm_knn <- confusionMatrix(data = y_hat_knn, reference = y_train, positive = '>50K')

print(cm_knn$table)
```

```
##            Reference
## Prediction  >50K <=50K
##      >50K   4545  2713
##      <=50K  4805 27011
```

```r
print(cm_knn$overall)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.075958e-01   4.276131e-01   8.036518e-01   8.114942e-01   7.607104e-01
## AccuracyPValue  McnemarPValue
##  4.904238e-110  1.704051e-128
```

```r
print(cm_knn$byClass)
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.4860963            0.9087270            0.6262056
##       Neg Pred Value            Precision               Recall
##            0.8489754            0.6262056            0.4860963
##                   F1           Prevalence       Detection Rate
##            0.5473266            0.2392896            0.1163178
## Detection Prevalence    Balanced Accuracy
##            0.1857501            0.6974116
```

Part f (5 points)

Using the output from parts d and e, write a few sentences comparing and contrasting the strengths and weaknesses of each model when it comes to predictions.

Response: Whilst nearly equivalent in terms of *Accuracy* (kNN: 0.807, logistic: 0.802) for the training data, it appears that `logistic` regression has a marginal advantage in *Specificity* (0.936 vs. 0.909) and *Precision* (0.648 vs. 0.626) of predictions, whereas `k-NN` has an advantage in *Sensitivity/Recall* (0.486 vs. 0.374). Thus `k-NN` may more reliably predict *True Positive* outcomes, whereas `logistic` regression, in particular at the 0.50 LD cutoff, may more reliably avoid *False Positive* outcomes. It may be noteworthy that the joint-probability κ of `k-NN` is higher, perhaps suggesting greater reliability of classification.

Part g (8 points)

Using the `train()` function in the `caret` package, perform 5-fold cross validation for k in the kNN model using only the training set and again using `age`, `education`, `race`, `sex`, and `hours_per_week` as the predictors. Set the search for k to be from 1 to 21 (we'll stop at 21 to save time). Make sure to use the `trControl` option to set it to cross validation. Then use Cohen's κ to determine the best k value. You do not need to change the metric in the `train()` function. Just look at the output and select the k with the largest κ value.

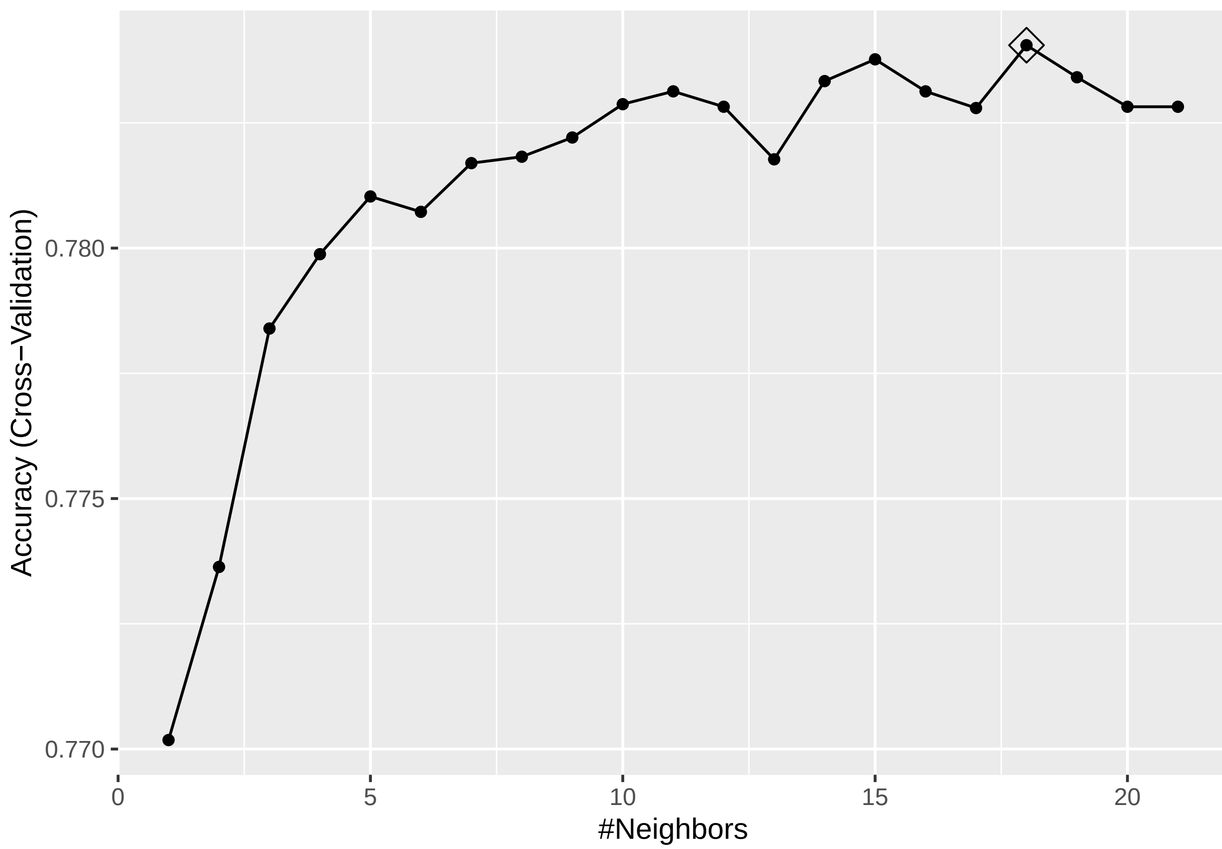
Then, if the best k is different than 7, fit another kNN model with the optimal k value. Please set a seed at the beginning of this code chunk.

It is fairly computationally expensive to optimize the k for the kNN model here since it takes so long to obtain the predictions. So, this may take a few minutes to run.

```
train_data <- adult_data |>
  slice(train_index)
train_data <- train_data |>
  select(age, education, race, sex, hrs_per_week, salary)

set.seed(2024)

train_knn <- train(salary ~ . , method = 'knn',
                   data = train_data,
                   trControl = trainControl(method ='cv'),
                   tuneGrid = data.frame( k = seq(1,21,1) )
                  )
ggplot(train_knn, highlight = TRUE)
```




```
train_knn$results
```

```
##      k  Accuracy     Kappa  AccuracySD     KappaSD
## 1    1 0.7701797 0.3347114 0.002915737 0.011884549
## 2    2 0.7736348 0.3397817 0.003641958 0.010459425
## 3    3 0.7783948 0.3486892 0.002892276 0.009620997
## 4    4 0.7798792 0.3527046 0.002154765 0.010835446
## 5    5 0.7810308 0.3538914 0.002167881 0.006112691
```

```
## 6   6 0.7807238 0.3518242 0.001766745 0.003994351
## 7   7 0.7816962 0.3546917 0.002776137 0.007341072
## 8   8 0.7818242 0.3535790 0.003257045 0.008492486
## 9   9 0.7822080 0.3545010 0.003674646 0.008564097
## 10 10 0.7828734 0.3556362 0.004486585 0.012522550
## 11 11 0.7831293 0.3557439 0.002983502 0.010921382
## 12 12 0.7828222 0.3537849 0.003200575 0.009647532
## 13 13 0.7817728 0.3503132 0.003625355 0.011660772
## 14 14 0.7833339 0.3526214 0.004311949 0.013747823
## 15 15 0.7837690 0.3520053 0.004320665 0.012870751
## 16 16 0.7831292 0.3489922 0.003961296 0.013376264
## 17 17 0.7827964 0.3472137 0.005182301 0.017660207
## 18 18 0.7840504 0.3487855 0.005452545 0.018975279
## 19 19 0.7834105 0.3452496 0.006396108 0.021824877
## 20 20 0.7828219 0.3423659 0.005731955 0.020401188
## 21 21 0.7828219 0.3407669 0.005821300 0.020743522
```

```r
train_k11 <- train(salary ~ . , method = 'knn', data = train_data,
                   tuneGrid = data.frame( k = 11 ))
```

Response: Highest $\kappa = 0.3557$ at $k = 11$, but highest *accuracy*$= 0.784$ at $k = 18$ with $\kappa = 0.34878$. Concluded by fitting a model with $k = 11$.

Part h (20 points)

We mentioned the most common cutoff value for the predicted probability for predicting a "success" in logistic regression is 0.5. However, we can adjust this value to make it easier or more difficult to predict a success. Let's optimize this cutoff value using 5-fold cross validation. Note: we could also do this with kNN, but we will not on this assignment.

Using the cutoff values from 0.15 to 0.85 by 0.05 (0.15, 0.20, 0.25, and so on up to 0.85) for predicting whether an adult has a salary above 50K, find which one performs best on the training set using the metric of Cohen's κ, which is given in the output of the `confusionMatrix()` function.

You will need a couple loops here since the `train()` function cannot do this for us. Note: you can find the indices of the rows in each fold using the `createFolds()` function in the `caret` library. Please set a seed at the beginning of your code chunk for this part.

```r
cutoff_values <- seq(0.15, 0.85, 0.05)

folds <- createFolds(train_data$salary, k = 5) #folds -- stratified sampling

fit_logit <- train(salary ~ ., method = 'glm', data = train_data,
             trControl = trainControl(method = 'cv', number = 5))


# store results
kappa_scores <- numeric( length(cutoff_values) )

# iterating through each cutoff value
for (i in seq_along(cutoff_values)) {

  # variable to store kappa scores for each fold
  kappa_scores_fold <- numeric(5)
```

```
  # iterate over folds
    for (fold in 1:5) {

      #probabilities for each fold
      fold_probs <- predict(fit_logit, type = 'prob', newdata = train_data[folds[[fold]], ])

      #need to effectively transpose 'fold_probs' with '>50K' column entries as "vector":
      fold_predictions <- factor(fold_probs[,2] > cutoff_values[i],
                                 levels = c(FALSE, TRUE),
                                 labels = c('<=50K', '>50K') )

      fold_cm <- confusionMatrix(data = fold_predictions,
                                  reference = train_data$salary[folds[[fold]]]
                                  )

      kappa_scores_fold[fold] <- fold_cm$overall['Kappa']

    }

    # other way of calculating Cohen's Kappa
    # kappa_scores_fold[fold] <- confusionMatrix(data = predictions, reference = valid_set$salary)$over

    # average kappa score across fold(s)
    kappa_scores[i] <- mean(kappa_scores_fold)

  }

optimal_cutoff <- cutoff_values[ which.max(kappa_scores) ]
optimal_cutoff
```

```
## [1] 0.35
```

Part i (5 points)

Once you have your "optimal" cutoff value, repeat part d using this cutoff and compare the results of this output to the results of the output for a kNN model with the optimal k value you found in part g. For which statistics is the logistic regression better now and for which is it worse?

```
y_hat_logit <- predict(logit_model, type='response')
#probabilities predicted from training data

#comparing with 0.35 'optimized' LD value
predicted_class <- factor(
  ifelse(y_hat_logit < 0.35, '>50K', '<=50K'), levels = c('>50K', '<=50K') #positive is '<=50K' because
  )

cm_logit <- confusionMatrix(data = predicted_class, reference = y_train, positive = '>50K')

print(cm_logit$table)
```

```
##             Reference
```

```
## Prediction  >50K <=50K
##      >50K   1877   718
##      <=50K  7473 29006
```

```r
print(cm_logit$overall)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   7.903721e-01   2.347068e-01   7.863019e-01   7.943992e-01   7.607104e-01
## AccuracyPValue  McnemarPValue
##   1.728122e-44   0.000000e+00
```

```r
print(cm_logit$byClass)
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.20074866           0.97584444           0.72331407
##       Neg Pred Value            Precision               Recall
##           0.79514241           0.72331407           0.20074866
##                   F1           Prevalence       Detection Rate
##           0.31427375           0.23928955           0.04803706
## Detection Prevalence    Balanced Accuracy
##           0.06641245           0.58829655
```

Part j (15 points)

Finally, let's test our two models (the logistic model with the "optimal" cutoff and the kNN model with the "optimal" k) on the test set. We must keep a few things in mind:

1. We must use the exact models we fit to the training set. You fit the logistic regression model in part c and you fit the kNN model in either part c or part g.
2. We should not use the results of the testing predictions to change our models. That should have been done with the training sets.

Find the predictions for the test set using the models, print the output of the `confusionMatrix()` function for each model, and compare the results in a few sentences.

```
##           Reference
## Prediction >50K <=50K
##      >50K   911   520
##      <=50K 1426  6911
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.007781e-01   3.688508e-01   7.927169e-01   8.086595e-01   7.607494e-01
## AccuracyPValue  McnemarPValue
##   1.652408e-21   1.573044e-93
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.38981600           0.93002288           0.63661775
##       Neg Pred Value            Precision               Recall
##           0.82895526           0.63661775           0.38981600
##                   F1           Prevalence       Detection Rate
##           0.48354565           0.23925061           0.09326372
## Detection Prevalence    Balanced Accuracy
##           0.14649877           0.65991944
```

```
##           Reference
## Prediction >50K <=50K
##      >50K   489   203
##      <=50K 1848  7228
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   7.900287e-01   2.397726e-01   7.818146e-01   7.980694e-01   7.607494e-01
## AccuracyPValue  McnemarPValue
##   3.215383e-12  1.558331e-288
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.20924262           0.97268201           0.70664740
##       Neg Pred Value            Precision               Recall
##           0.79638607           0.70664740           0.20924262
##                   F1           Prevalence       Detection Rate
##           0.32287884           0.23925061           0.05006143
## Detection Prevalence    Balanced Accuracy
##           0.07084357           0.59096231
```

```
## Warning in confusionMatrix.default(data = y_hat_knn_test, reference = y_test, :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
##           Reference
## Prediction >50K <=50K
##      >50K  1042   782
##      <=50K 1295  6649
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   7.873669e-01   3.683501e-01   7.791164e-01   7.954458e-01   7.607494e-01
## AccuracyPValue  McnemarPValue
##   2.257428e-10   2.762169e-29
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.4458708            0.8947652            0.5712719
##       Neg Pred Value            Precision               Recall
##            0.8369839            0.5712719            0.4458708
##                   F1           Prevalence       Detection Rate
##            0.5008411            0.2392506            0.1066749
## Detection Prevalence    Balanced Accuracy
##            0.1867322            0.6703180
```

Response: The logistic regression model with optimized $LD = 0.35$ cutoff very marginally outperforms the `kNN` model at $k = 11$ in terms of *Accuracy* with `kNN` accuracy = 0.7895 and `logit` accuracy = 0.7900. `Logistic` also outperforms in *Specificity* = 0.973 and *Precision* = 0.707 by good margins versus = 0.896 and = 0.577 respectively for the `kNN` model. From the confusion matrix, it can be seen that the `kNN` model accurately predicts *True Positives* more often compared to the logistic regression which appears to make many *True Negative* predictions in compensation.

Part k (5 points)

Even though one method may be better on a given dataset than another, that does not mean that method will always predict better. However, logistic regression has a few advantages over kNN regardless of predictive power. List at least three advantages logistic regression has over kNN.

Response:. Irrespective of context-superior predictive power, logistic regression is on a practical level less computationally expensive: it operates on linear relationships between the feature and outcome and learned coefficients. `kNN` requires iterative distance calculations between each data point and all other points, and is comparable to `k-Means` in this regard. Logistic regression also provides more understandable interpretability as the coefficients are signed magnitudes of each predictor variable, applied by the logistic/link function. `kNN` classes observations based on 'votes' from the k-neighboring data points which may be difficult to interpret in particular with hyper-dimensional data. Because of logistic regression's relatively simple calculation, linear combinations of parameters applied by the logistic function, it has less likelihood of overfitting. `kNN`, as an iterative comparison of $\frac{n*(n-1)}{2}$ distances will learn noise in the data, leading to overfitting and less generalizability.