# 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 under 4 hours.

$$\frac{58}{414} + \frac{37}{414} = \frac{95}{414} = 0.229$$

- The overall accuracy of these predictions.

$$\frac{58}{414} + \frac{217}{414} = \frac{275}{414} = 0.664$$

- The sensitivity (recall).

$$\frac{58}{95} = 0.611$$

- The specificity.

$$\frac{217}{319} = 0.680$$

- The positive predictive value (precision).

$$\frac{58}{160} = 0.363$$

- The negative predictive value.
$$\frac{217}{254} = 0.854$$
- The balanced accuracy.
$$\frac{(0.611 + 0.680)}{2} = 0.6455$$
- Cohen's Kappa ($\kappa$). Check out this link on Wikipedia and scroll down to the section entitled **Binary classification confusion matrix**.
$$\frac{2(58 * 217 - 37 * 102)}{(28 + 102) * (102 + 217) + (58 + 37) * (37 * 217)} = 0.234$$

**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.

# 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.

```
adult <-  read_csv("adult.data", col_names = FALSE)|>
  rename("age" = "X1", "wClass" = "X2", "fnlwgt" = "X3",
              "education" = "X4", "education-num" = "X5", "mStatus" = "X6",
              "occup" = "X7", "relationship" = "X8", "race" = "X9",
              "sex" = "X10", "capGain" = "X11", "capLoss" = "X12",
              "hours_per_week" = "X13", "nCountry" = "X14",
              "salary" = "X15") |>
  tibble()
```

```
## 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.
```

```
adult$salary = as_factor(adult$salary)
```

### 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.

```
set.seed(1)
testIndex <- createDataPartition(adult$salary, times = 1, p = 0.8,
                                 list = F)
train <- adult[testIndex, ]
test <- adult[-testIndex, ]
```

### 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.

```
lgMod <- glm(salary ~ age + education + race + sex + hours_per_week,
             family = 'binomial', data = train)
```

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.

```
kn <- knn3(salary ~ age + education + race + sex + hours_per_week,
           data = train, k = 7)
```

### 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.

```
y = predict(lgMod, train, type = "response")
y_hat <- ifelse(y > .5, ">50K", "<=50K") |>
  factor(levels = levels(adult$salary))

lgCm <- confusionMatrix(data = y_hat, reference = train$salary)
print(lgCm)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction <=50K  >50K
```

```
##      <=50K 27808  5848
##      >50K   1916  3502
##
##                Accuracy : 0.8013
##                  95% CI : (0.7973, 0.8052)
##     No Information Rate : 0.7607
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3623
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9355
##             Specificity : 0.3745
##          Pos Pred Value : 0.8262
##          Neg Pred Value : 0.6464
##              Prevalence : 0.7607
##          Detection Rate : 0.7117
##    Detection Prevalence : 0.8613
##       Balanced Accuracy : 0.6550
##
##        'Positive' Class : <=50K
##
```

**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.

```
kNy_hat = predict(kn, train, type = "class")

knCm <- confusionMatrix(data = kNy_hat, reference = train$salary)
print(knCm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K  >50K
##      <=50K 27461  5167
##      >50K   2263  4183
##
##                Accuracy : 0.8098
##                  95% CI : (0.8059, 0.8137)
##     No Information Rate : 0.7607
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4155
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9239
##             Specificity : 0.4474
##          Pos Pred Value : 0.8416
```

```
##          Neg Pred Value : 0.6489
##              Prevalence : 0.7607
##          Detection Rate : 0.7028
##    Detection Prevalence : 0.8350
##       Balanced Accuracy : 0.6856
##
##        'Positive' Class : <=50K
##
```

### 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.

**Overall accuracy on both models doesn't look too bad. On the logistic model, specificity is absolutely terrible, which affected the balanced accuracy a lot. Some strengths of this model are the sensitivity, positive prediction, and detection prevalence. So it seems like it detects those who make <=50k decently well, at the expense of predicting negatives. The kappa is also rough. The k-NN seems to struggle from a lot of the same, although the kappa is better. Overall, they seem to perform pretty similarly which is a bit concerning and makes me feel like I did this wrong.**

### 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 $\kappa$ 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 $\kappa$ 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.

```
set.seed(1)

knTrain <- train(salary ~ age + education + race + sex + hours_per_week,
                 method = "knn",
                 data = train,
                 trControl = trainControl(method = "cv", number = 5),
                 tuneGrid = data.frame(k = seq(1, 21, 1)))
sort(knTrain[["results"]][["Kappa"]])
```

```
##  [1] 0.3066822 0.3099526 0.3100935 0.3109886 0.3142945 0.3149029 0.3156719
##  [8] 0.3186145 0.3218513 0.3230957 0.3231265 0.3239733 0.3285822 0.3287411
## [15] 0.3302033 0.3302326 0.3320783 0.3332171 0.3334108 0.3341602 0.3346765
```

```
kn5 <- knn3(salary ~ age + education + race + sex + hours_per_week,
           data = train, k = 5)
kNy_hat5 = predict(kn5, train, type = "class")
```

```
knCm5 <- confusionMatrix(data = kNy_hat5, reference = train$salary)
print(knCm5)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K  >50K
##      <=50K 27507  5028
##      >50K   2217  4322
##
##                Accuracy : 0.8146
##                  95% CI : (0.8107, 0.8184)
##     No Information Rate : 0.7607
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4322
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9254
##             Specificity : 0.4622
##          Pos Pred Value : 0.8455
##          Neg Pred Value : 0.6610
##              Prevalence : 0.7607
##          Detection Rate : 0.7040
##    Detection Prevalence : 0.8327
##       Balanced Accuracy : 0.6938
##
##        'Positive' Class : <=50K
##
```

### 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 $\kappa$, 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.

```
set.seed(1)
cutoffs <- seq(0.15, 0.85, 0.05)
folds <- createFolds(train$salary, k = 5)

to_vec(for(p in cutoffs) {
    to_vec(for(i in c("Fold1", "Fold2", "Fold3", "Fold4", "Fold5")) {

      testSet <-  train[folds[[i]],]
```

```
      #test set made of current fold
      trainSet <-  train[- folds[[i]],]
      #train set made of everything that is not current fold
      mod <-  glm(salary ~ age + education + race + sex +
                    hours_per_week,
                  family = 'binomial', data = trainSet)

      y = predict(mod, testSet, type = "response")
      #get predictions of models

      y_hat <- ifelse(y > p, ">50K", "<=50K") |>
      #classify predictions based on  cutoff
      factor(levels = levels(adult$salary))

      cm <- confusionMatrix(data = y_hat, reference = testSet$salary)
      kappa <- cm$overall[["Kappa"]]
      kappa
  }) |> mean()
}) -> kappas
```

```
cutoffs[which(kappas == max(kappas))]
```

```
## [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 = predict(lgMod, train, type = "response")
y_hat <- ifelse(y > .35, ">50K", "<=50K") |>
  factor(levels = levels(adult$salary))

lgCm <- confusionMatrix(data = y_hat, reference = train$salary)
print(lgCm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K  >50K
##      <=50K 25250  3930
##      >50K   4474  5420
##
##                Accuracy : 0.7849
##                  95% CI : (0.7808, 0.789)
##     No Information Rate : 0.7607
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4208
##
```

```
##  Mcnemar's Test P-Value : 3.157e-09
##
##             Sensitivity : 0.8495
##             Specificity : 0.5797
##          Pos Pred Value : 0.8653
##          Neg Pred Value : 0.5478
##              Prevalence : 0.7607
##          Detection Rate : 0.6462
##    Detection Prevalence : 0.7468
##       Balanced Accuracy : 0.7146
##
##        'Positive' Class : <=50K
##
```

**The logistic regression has a better sensitivity, but pretty much every other statistic was worse.**

### 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.

```r
lgTest <- glm(salary ~ age + education + race + sex + hours_per_week,
              family = 'binomial', data = test)

yTest <-  predict(lgTest, test, type = "response")

y_hatTest <- ifelse(yTest > .35, ">50K", "<=50K") |>
  factor(levels = levels(adult$salary))

lgCmTest <- confusionMatrix(data = y_hatTest, reference = test$salary)
print(lgCmTest)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K >50K
##      <=50K  6334  990
##      >50K   1097 1347
##
##                Accuracy : 0.7863
##                  95% CI : (0.7781, 0.7944)
##     No Information Rate : 0.7607
##     P-Value [Acc > NIR] : 1.04e-09
##
```

```
##                   Kappa : 0.4221
##
##  Mcnemar's Test P-Value : 0.02032
##
##             Sensitivity : 0.8524
##             Specificity : 0.5764
##          Pos Pred Value : 0.8648
##          Neg Pred Value : 0.5511
##              Prevalence : 0.7607
##          Detection Rate : 0.6484
##    Detection Prevalence : 0.7498
##       Balanced Accuracy : 0.7144
##
##        'Positive' Class : <=50K
##
```

```
knTest <- knn3(salary ~ age + education + race + sex + hours_per_week,
           data = test, k = 5)
kNy_test = predict(knTest, test, type = "class")

knCmTest <- confusionMatrix(data = kNy_test, reference = test$salary)
print(knCmTest)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K >50K
##      <=50K  6887 1211
##      >50K    544 1126
##
##                Accuracy : 0.8203
##                  95% CI : (0.8126, 0.8279)
##     No Information Rate : 0.7607
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4529
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9268
##             Specificity : 0.4818
##          Pos Pred Value : 0.8505
##          Neg Pred Value : 0.6743
##              Prevalence : 0.7607
##          Detection Rate : 0.7051
##    Detection Prevalence : 0.8290
##       Balanced Accuracy : 0.7043
##
##        'Positive' Class : <=50K
##
```

**The kappa value for the kNN model is slightly better than the logistic. The kNN model is also more accurate and has better sensitivity. The logistic model has better specificity and**

**positive prediction. In fact, the specificity has greatly improved, indicating the logistic model now does a better job at predicting negatives than it did before. The balanced accuracy has also greatly improved for the logistic model, likely because the specificity is so much better**

**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.

**1. Logistic regression gives the probability of an observation belonging to a given class, so the cutoff can be adjusted.**

**2. Logistic regression is less sensitive to outliers**

**3. Logistic regression is faster**