```r
# AD 699 Assignment 3
# Tiange Chang

library(tidyverse)

# Classification

# 1
setwd('/Users/t/Desktop')
songs <- read_csv("top2020_21.csv")
View(songs)

# 1. a
# I picked h u n g e r . o n . h i l l s i d e (with Bas) by J. Cole.

# 1. b
# J. Cole is one of my favorite rapper, this song is from his newest album.
# I like the beat of this song, it sampled another song called "I wonder where
# our love has gone" by Junior Parker.

# 1. c
# danceability: 0.563
# energy: 0.703
# loudness: 5.756
# speechiness: 0.0702
# acousticness: 0.126
# liveness: 0.213
# valence: 0.294
# tempo: 94.983
# duration_ms: 238972

# 2
names(songs) <- str_replace_all(names(songs), c(" " = "_"))
View(songs)

mysong <- subset(songs, Song_Name ==
                   'h u n g e r . o n . h i l l s i d e (with Bas)')
View(mysong)
```

699 A3.R* × | mysong ×

Filter

	Index	Highest_Charting_Position	Number_of_Times_Charted	Week_of_Highest_Charting	Song_Name	Streams	Artist	Artist_Followers
1	363	48	1	2021-05-14--2021-05-21	h u n g e r . o n . h i l l s i d e (with Bas)	11647489	J. Cole	14097410

```r
# 3. a
# from str(sptfy), we can know that the type of 'target' is num.
sptfy$target <- as.factor(sptfy$target)
str(sptfy)
# now the type of 'target' is Factor
View(sptfy)
```

```r
# 3. b
# The target variable has two unique values: 1 and 0.
table(sptfy$target)
# From the table function, we can know that there are 997 songs George disliked,
# and 1020 songs George liked.
```

```r
# 4
any(is.na(sptfy))
# Unfortunately, there is no NAs in the dataset.

# 5
sptfy = select(sptfy, -1, -10)
View(sptfy)

# 6
set.seed(20)
train <- sample_frac(sptfy, 0.6)
valid <- setdiff(sptfy, train)

# 7. a
copy <- train
```

```r
# 7. b
# songs George like
G1 <- copy %>% filter(target==1) %>% select( 1, 2, 3, 4, 6, 7, 8, 9, 10, 12)
View(G1)

# songs George dislike
G0 <- copy %>% filter(target==0) %>% select( 1, 2, 3, 4, 6, 7, 8, 9, 10, 12)
View(G0)

m1 <- colMeans(G1)
m0 <- colMeans(G0)

# percentage difference in mean
PD <- (abs(m1 - m0) / ((m1 + m0) / 2)) * 100
PD
```

```
> PD
acousticness danceability  duration_ms       energy          key     liveness     loudness  speechiness        tempo      valence
   39.447886     8.234466    11.211090     2.415148     3.543574     5.418185    -7.385134    24.356442     1.068906    10.779291
```

```r
# 7. c
# Variables acousticness, danceability, duration_ms, liveness, loudness,
# speechiness, and valence show a percentage difference of 5% or more.
# Hence, we are going to remove energy, key, tempo.

# Drop variables from copy
G1 <- select(G1, -4, -5, -9)
View(G1)

G0 <- select(G0, -4, -5, -9)
View(G1)

# Drop variables from train & valid
View(train)

train <- select(train, -4, -6, -10)
View(train)

valid <- select(valid, -4, -6, -10)
View(valid)
```

```r
# 7. d
# In k-nn model, when the variables' values are very similar for both outcome
# classes, it could lead to overfitting. That is why we remove those variables.
```

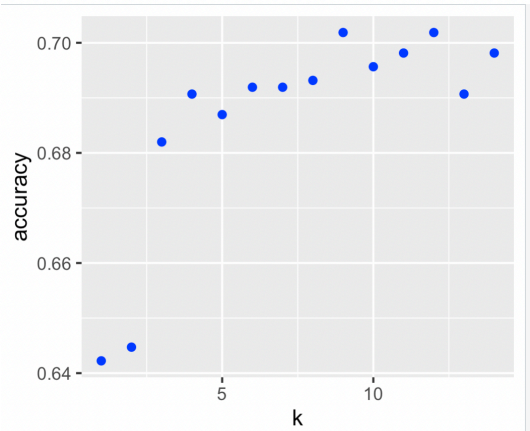
```r
# 8
library(caret)

View(sptfy)
sptfy2 <- select(sptfy, -4, -6, -10)
View(sptfy2)

# Initialize normalized training, validation data, complete df to originals
train.norm <- train
valid.norm <- valid
sptfy.norm <- sptfy2
new <- data.frame(acousticness = 0.126, danceability = 0.563,
                  duration_ms = 238972, liveness = 0.213,
                  loudness = -5.756, speechiness = 0.0702,
                  valence = 0.294)
View(new)

# preProcess()
norm.values <- preProcess(train[, c(1, 2, 3, 5, 6, 7, 9)],
                          method = c("center", "scale"))

train.norm[, c(1, 2, 3, 5, 6, 7, 9)] <- predict(
  norm.values, train[, c(1, 2, 3, 5, 6, 7, 9)])

valid.norm[, c(1, 2, 3, 5, 6, 7, 9)] <- predict(
  norm.values, valid[, c(1, 2, 3, 5, 6, 7, 9)])

sptfy.norm[, c(1, 2, 3, 5, 6, 7, 9)] <- predict(
  norm.values, sptfy2[, c(1, 2, 3, 5, 6, 7, 9)])

new.norm <- predict(norm.values, new)
> new.norm
  acousticness danceability duration_ms  liveness  loudness speechiness    valence
1   -0.2378398   -0.3087532 -0.08139447 0.1458892 0.3415035   -0.235108 -0.8078656
```

```
# 9
library(FNN)

nn <- knn(train = train.norm[, c(1, 2, 3, 5, 6, 7, 9)], test = new.norm,
          cl = train.norm[, 10, drop = TRUE], k = 7)
nn

row.names(train)[attr(nn, "nn.index")]

# Predicted classification for my song
PD2 <- sptfy2[c(158, 327, 25, 626, 750, 459, 285), 10:12]
PD2

# I think George will like it, all seven songs have the target of one. The
# top one song predicted was Alright by Kendrick Lamar. J. Cole and Kendrick
# are known as two generational talent rappers. They both wrote legit lyrics.
# The outcomes are shown below.
```

```
> nn <- knn(train = train.norm[, c(1, 2, 3, 5, 6, 7, 9)], test = new.norm,
+           cl = train.norm[, 10, drop = TRUE], k = 7)
> nn
[1] 1
attr(,"nn.index")
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]  158  327   25  626  750  459  285
attr(,"nn.dist")
          [,1]      [,2]      [,3]      [,4]      [,5]     [,6]      [,7]
[1,] 0.4934895 0.5158846 0.6103108 0.6755426 0.7060524 0.760613 0.7685626
Levels: 1
>
> row.names(train)[attr(nn, "nn.index")]
[1] "158" "327" "25"  "626" "750" "459" "285"
>
> # Predicted classification for my song
> PD2 <- sptfy2[c(158, 327, 25, 626, 750, 459, 285), 10:12]
> PD2
# A tibble: 7 × 3
  target song_title            artist
  <fct>  <chr>                 <chr>
1 1      Alright               Kendrick Lamar
2 1      Hands On The Wheel    ScHoolboy Q
3 1      If I Gave You My Love Myron & E
4 1      Had A Dream           EMP DASME
5 1      Viol - Original Mix   Gesaffelstein
6 1      Osaka Loop Line       Discovery
7 1      Archangel             Burial
```

```r
# 10
train.norm$target <- as.factor(train.norm$target)
valid.norm$target <- as.factor(valid.norm$target)

str(train.norm)
str(valid.norm)

accuracy <- data.frame(k = seq(1, 14, 1), accuracy = rep(0, 14))

for(i in 1:14) {
  knn.pred <- knn(train.norm[, c(1, 2, 3, 5, 6, 7, 9)],
                  valid.norm[, c(1, 2, 3, 5, 6, 7, 9)],
                  cl = train.norm[, 10, drop = TRUE], k = i)
  accuracy[i, 2] <- confusionMatrix(knn.pred,
                                    valid.norm[, 10, drop = TRUE])$overall[1]
}

accuracy
# The optimal k-value is 12, with the accuracy of 0.7018634.
```

```
> accuracy
    k  accuracy
1   1 0.6422360
2   2 0.6447205
3   3 0.6819876
4   4 0.6906832
5   5 0.6869565
6   6 0.6919255
7   7 0.6919255
8   8 0.6931677
9   9 0.7018634
10 10 0.6956522
11 11 0.6981366
12 12 0.7018634
13 13 0.6906832
14 14 0.6981366
```




```r
# 11
ggplot(data = accuracy, aes(x=k, y=accuracy)) + geom_point(color = "blue")
```

```r
# 12
nn <- knn(train = train.norm[, c(1, 2, 3, 5, 6, 7, 9)], test = new.norm,
          cl = train.norm[, 10, drop = TRUE], k = 12)
nn

row.names(train)[attr(nn, "nn.index")]

# Predicted classification for my song
PD3 <- sptfy2[c(158, 327, 25, 626, 750, 459, 285, 114, 756, 864, 364, 1026),
              10:12]
PD3
# The result is the same, George will like the song. The top one result has
# value of 0.4934895. And 11 out of 12 songs have target of one.
```

```
> nn <- knn(train = train.norm[, c(1, 2, 3, 5, 6, 7, 9)], test = new.norm,
+           cl = train.norm[, 10, drop = TRUE], k = 12)
> nn
[1] 0
attr(,"nn.index")
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]  158  327   25  626  750  459  285  114  756   864   364  1026
attr(,"nn.dist")
          [,1]      [,2]      [,3]      [,4]      [,5]     [,6]      [,7]      [,8]      [,9]
[1,] 0.4934895 0.5158846 0.6103108 0.6755426 0.7060524 0.760613 0.7685626 0.7968182 0.8061877
         [,10]    [,11]     [,12]
[1,] 0.8100065 0.815839 0.8186296
Levels: 0
>
> row.names(train)[attr(nn, "nn.index")]
 [1] "158"  "327"  "25"   "626"  "750"  "459"  "285"  "114"  "756"  "864"  "364"  "1026"
>
> # Predicted classification for my song
> PD3 <- sptfy2[c(158, 327, 25, 626, 750, 459, 285, 114, 756, 864, 364, 1026),
+               10:12]
> PD3
# A tibble: 12 × 3
   target song_title            artist
   <fct>  <chr>                 <chr>
 1 1      Alright               Kendrick Lamar
 2 1      Hands On The Wheel    ScHoolboy Q
 3 1      If I Gave You My Love Myron & E
 4 1      Had A Dream           EMP DASME
 5 1      Viol - Original Mix   Gesaffelstein
 6 1      Osaka Loop Line       Discovery
 7 1      Archangel             Burial
 8 1      Fiesta - Remix        Bomba Estéreo
 9 1      Collard Greens        ScHoolboy Q
10 1      5thep                 Todd Osborn
11 1      Sick Beat             Kero Kero Bonito
12 0      More Girls Like You   Kip Moore
```

```r
# Naive Bayes

install.packages('fivethirtyeight')
library(fivethirtyeight)

# 1
data('weather_check')
wc <- weather_check

View(wc)
str(wc)
```

```r
# 2. a
fct_count(wc$weather_source)
# From the function, we can know that there are 9 levels: A specific website
# or app, Internet search, Local TV News, Newsletter, Newspaper, Radio weather
# The default weather app on your phone, The Weather Channel, and NAs. Our goal
# is to reduce that level to five.

# I want to do five levels:
# 1: Newsletter & Newspaper & Radio weather & Local TV News -> Old_Fashioned
# 2: Internet search & The default weather app on your phone  -> Modern
# 3: A specific website or app -> Specific
# 4: The Weather Channel -> Channel
# 5: NA -> NA

wc$weather_source <- fct_collapse(wc$weather_source,
  Old_Fashioned = c('Newsletter', 'Newspaper',
                    'Radio weather', 'Local TV News'),
         Modern = c('Internet search', 'The default weather app on your phone'),
       Specific = c('A specific website or app (please provide the answer)'),
        Channel = c('The Weather Channel'))

fct_count(wc$weather_source)
```

	f	n
	<fct>	<int>
1	Specific	175
2	Modern	343
3	Old_Fashioned	260
4	Channel	139
5	NA	11

```r
# 2. b
str(wc)
# From the function, we can know that weather_source_site and region are chr.

# Convert to factors
wc$weather_source_site <- factor(wc$weather_source_site)
wc$region <- factor(wc$region)
str(wc)
```

```
> str(wc)
tibble [928 × 7] (S3: tbl_df/tbl/data.frame)
 $ ck_weather      : logi [1:928] TRUE TRUE TRUE TRUE TRUE TRUE ...
 $ weather_source  : Factor w/ 4 levels "Specific","Modern",..: 2 2 2 2 1 1 4 NA 4 2 ...
 $ ck_weather_watch: Ord.factor w/ 4 levels "Very unlikely"<..: 4 4 4 3 4 3 1 NA 4 4 ...
 $ age             : Factor w/ 4 levels "18 - 29","30 - 44",..: 2 1 2 2 2 1 2 NA 2 2 ...
 $ female          : logi [1:928] FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ hhold income    : Ord.factor w/ 11 levels "$0 to $9,999"<..: 4 11 6 11 8 6 3 NA 11 8 ...
```

```r
# 2. c
install.packages('questionr')
library(questionr)

anyNA(wc)
freq.na(wc)
# From the function, we can know that there are missing values in the dataset.
# The table is shown below.

# 2. c. i
# I think the presence of an NA can indicate something about the observation.
# "The cause of the presence of missing values in the dataset can be loss of
# information, disagreement in uploading the data, and many more." According
# to Satyam Kumar(01/31/2021).

# Reference
# https://towardsdatascience.com/deep-dive-analysis-of-missing-values-in-
# dataset-b387d9de6d4b#:~:text=The%20real%2Dworld%20dataset%20often,of%20the%
# 20model%20development%20pipeline.

# 2. c. ii
# The weather_source_site has the high degree of missingness of 81%.

table(wc$weather_source_site)
View(wc)

# Usually, we can use the mean to predict the missing value, but we cannot
# predict URLs, it is very unique. So it is a good idea to just remove it!

wc <- select(wc, -4)
View(wc)

freq.na(wc)
```

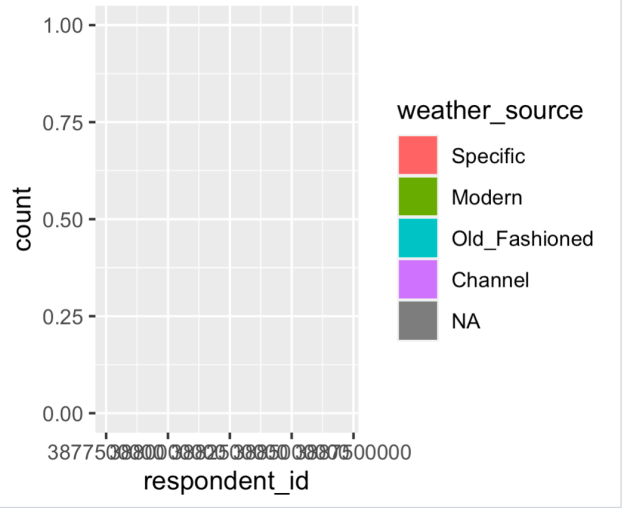
```
> freq.na(wc)
                      missing  %
weather_source_site       753 81
region                     31  3
hhold_income               13  1
age                        12  1
female                     12  1
weather_source             11  1
ck_weather_watch           11  1
respondent_id               0  0
ck_weather                  0  0
```

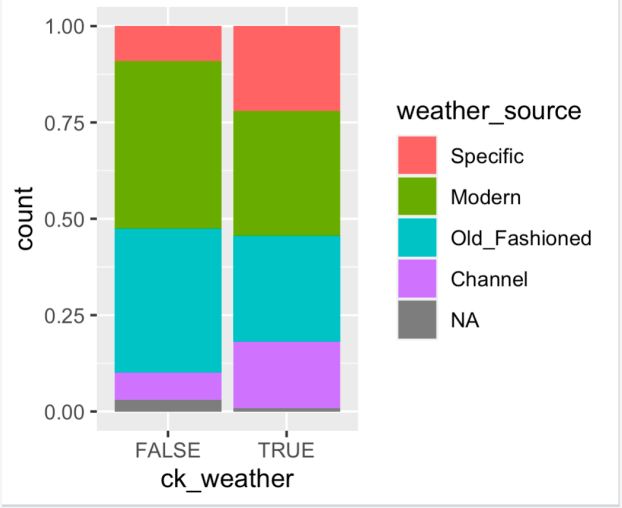
```
> table(wc$weather_source_site)

                                          1 weather
                                                  1
                                           1weather
                                                  1
                                        accuweather
                                                  5
                                        Accuweather
                                                 10
                                        AccuWeather
                                                  2
                                    AccuWeather App
                                                  1
                        AccuWeather or Weather Underground
                                                  1
                                                aol
                                                  1
                                    App on Iphone
                                                  1
                                     Apple weater
                                                  1
                                Apple Weather App
                                                  1
                            apple-provided site
                                                  1
                basic weather app on my iPhone
                                                  1
                                      Bing
                                        1
                                    Chrome app
                                                  1
```

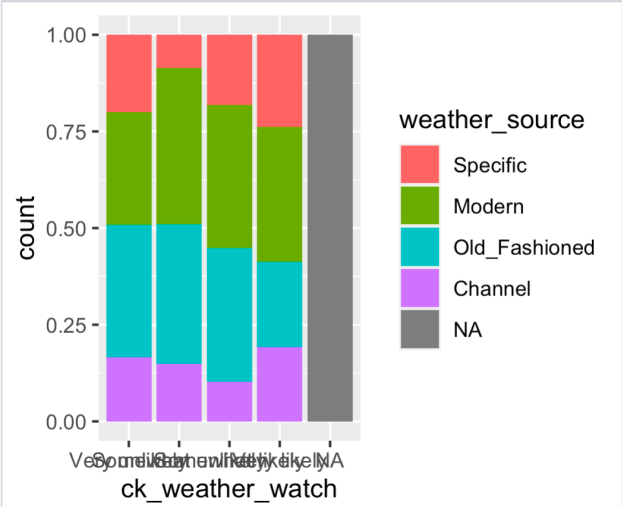
```
> freq.na(wc)
                      missing %
region                     31 3
hhold_income               13 1
age                        12 1
female                     12 1
weather_source             11 1
ck_weather_watch           11 1
respondent_id               0 0
ck_weather                  0 0
```

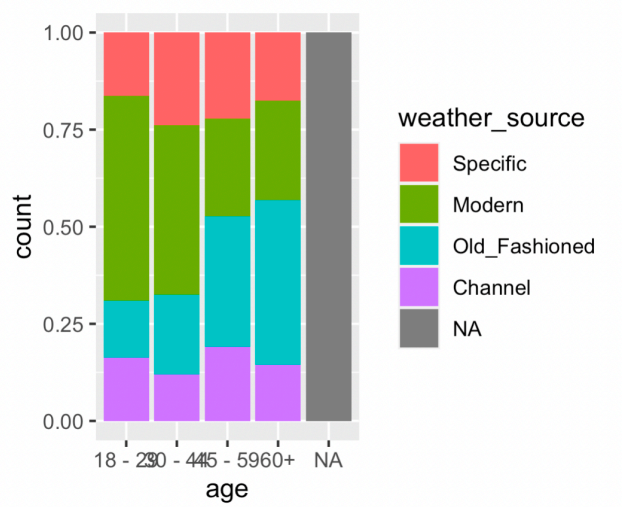
```r
# 3
set.seed(20)
train.wc <- sample_frac(wc, 0.6)
valid.wc <- setdiff(wc, train.wc)
```

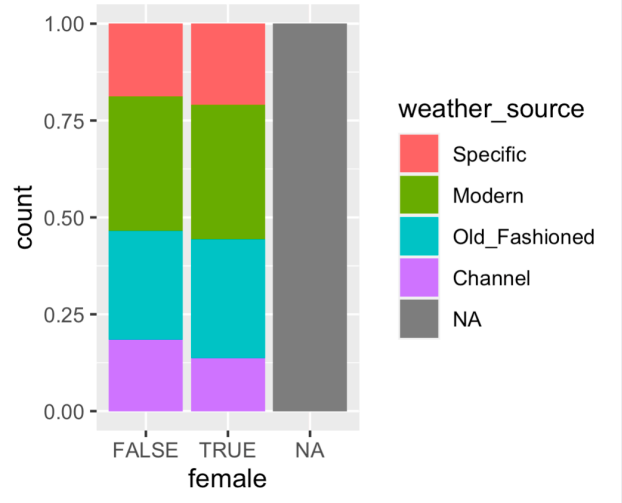
```r
# 4. a
names(wc)
# There are 8 variables.

ggplot(data = train.wc, aes(x = respondent_id, fill = weather_source)) +
  geom_bar(position = 'fill')

ggplot(data = train.wc, aes(x = ck_weather, fill = weather_source)) +
  geom_bar(position = 'fill')

ggplot(data = train.wc, aes(x = ck_weather_watch, fill = weather_source)) +
  geom_bar(position = 'fill')

ggplot(data = train.wc, aes(x = age, fill = weather_source)) +
  geom_bar(position = 'fill')

ggplot(data = train.wc, aes(x = female, fill = weather_source)) +
  geom_bar(position = 'fill')

ggplot(data = train.wc, aes(x=hhold_income, fill = weather_source)) +
  geom_bar(position = 'fill')

ggplot(data = train.wc, aes(x = region, fill = weather_source)) +
  geom_bar(position = 'fill')
```

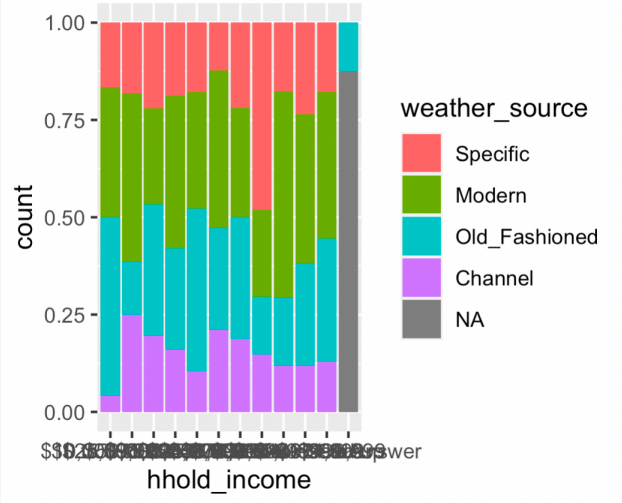


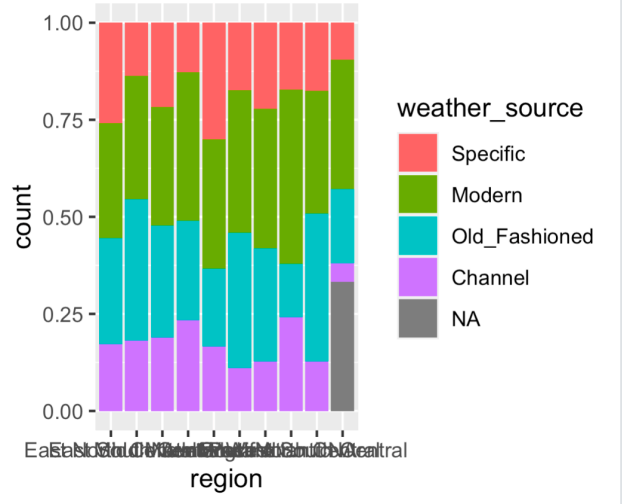













```
# 4. b
# respondent_id seems like won't have any predictive power in this model.
```

```r
# 4. b. i
View(train.wc)
train.wc <- select(train.wc, -1)
View(train.wc)
wc <- select(wc, -1)
valid.wc <- setdiff(wc, train.wc)
# 5
install.packages('e1071')
library(e1071)

model <- naiveBayes(weather_source~., data = train.wc)
model
```

```
> model

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
     Specific        Modern Old_Fashioned       Channel
    0.2000000     0.3472727     0.2963636     0.1563636

Conditional probabilities:
               ck_weather
Y                    FALSE       TRUE
  Specific      0.08181818 0.91818182
  Modern        0.22513089 0.77486911
  Old_Fashioned 0.22699387 0.77300613
  Channel       0.08139535 0.91860465

               ck_weather_watch
Y               Very unlikely Somewhat unlikely Somewhat likely Very likely
  Specific         0.21818182        0.03636364      0.27272727  0.47272727
  Modern           0.18324607        0.09947644      0.31937173  0.39790576
  Old_Fashioned    0.25153374        0.10429448      0.34969325  0.29447853
  Channel          0.23255814        0.08139535      0.19767442  0.48837209

               age
Y                  18 - 29    30 - 44    45 - 59        60+
  Specific      0.16363636 0.25454545 0.32727273 0.25454545
  Modern        0.30366492 0.26701571 0.21465969 0.21465969
  Old_Fashioned 0.09815951 0.14723926 0.33742331 0.41717791
  Channel       0.20930233 0.16279070 0.36046512 0.26744186

               female
Y                    FALSE      TRUE
  Specific      0.4000000 0.6000000
  Modern        0.4240838 0.5759162
  Old_Fashioned 0.4049080 0.5950920
  Channel       0.5000000 0.5000000
```

```
              hhold_income
Y              $0 to $9,999 $10,000 to $24,999 $25,000 to $49,999 $50,000 to $74,999
  Specific        0.03636364         0.07272727         0.15454545         0.11818182
  Modern          0.04188482         0.09947644         0.09947644         0.14136126
  Old_Fashioned   0.06790123         0.03703704         0.16049383         0.11111111
  Channel         0.01162791         0.12790698         0.17441860         0.12790698
              hhold_income
Y              $75,000 to $99,999 $100,000 to $124,999 $125,000 to $149,999 $150,000 to $174,999
  Specific              0.10909091           0.06363636           0.06363636           0.11818182
  Modern                0.10471204           0.12041885           0.04712042           0.03141361
  Old_Fashioned         0.17283951           0.09259259           0.06172840           0.02469136
  Channel               0.08139535           0.13953488           0.06976744           0.04651163
              hhold_income
Y              $175,000 to $199,999 $200,000 and up Prefer not to answer
  Specific                0.02727273      0.07272727           0.16363636
  Modern                  0.04712042      0.06806283           0.19895288
  Old_Fashioned           0.01851852      0.05555556           0.19753086
  Channel                 0.02325581      0.04651163           0.15116279

              region
Y              East North Central East South Central Middle Atlantic   Mountain New England
  Specific              0.19444444         0.02777778      0.13888889 0.05555556  0.08333333
  Modern                0.13043478         0.03804348      0.11413043 0.09782609  0.05434783
  Old_Fashioned         0.13836478         0.05031447      0.12578616 0.07547170  0.03773585
  Channel               0.16470588         0.04705882      0.15294118 0.12941176  0.05882353
              region
Y                 Pacific South Atlantic West North Central West South Central
  Specific      0.17592593     0.17592593         0.04629630         0.10185185
  Modern        0.21739130     0.16847826         0.07065217         0.10869565
  Old_Fashioned 0.23899371     0.15723270         0.02515723         0.15094340
  Channel       0.14117647     0.12941176         0.08235294         0.09411765
```

```r
# 6
PD3 <- predict(model, train.wc)
model.t <- confusionMatrix(PD3, train.wc$weather_source)
model.t

PD4 <- predict(model, valid.wc)
model.v <- confusionMatrix(PD4, valid.wc$weather_source)
model.v

# Train model accuracy is 0.4836, Valid model accuracy is 0.3711. The training
# set's performance is better than the validation set's performance.
```

```
> PD3 <- predict(model, train.wc)
> model.t <- confusionMatrix(PD3, train.wc$weather_source)
> model.t
Confusion Matrix and Statistics

              Reference
Prediction      Specific Modern Old_Fashioned Channel
  Specific            33     19            14      10
  Modern              40    113            38      35
  Old_Fashioned       31     50           105      26
  Channel              6      9             6      15

Overall Statistics

               Accuracy : 0.4836
                 95% CI : (0.4411, 0.5263)
    No Information Rate : 0.3473
    P-Value [Acc > NIR] : 3.388e-11

                  Kappa : 0.2678

 Mcnemar's Test P-Value : 6.167e-08

Statistics by Class:

                     Class: Specific Class: Modern Class: Old_Fashioned Class: Channel
Sensitivity                   0.3000        0.5916               0.6442        0.17442
Specificity                   0.9023        0.6852               0.7235        0.95474
Pos Pred Value                0.4342        0.5000               0.4953        0.41667
Neg Pred Value                0.8376        0.7593               0.8284        0.86187
Prevalence                    0.2000        0.3473               0.2964        0.15636
> PD4 <- predict(model, valid.wc)
> model.v <- confusionMatrix(PD4, valid.wc$weather_source)
> model.v
Confusion Matrix and Statistics

              Reference
Prediction      Specific Modern Old_Fashioned Channel
  Specific             6     15             9       3
  Modern              25     67            35      24
  Old_Fashioned       20     32            41      17
  Channel              4     11             5       4

Overall Statistics

               Accuracy : 0.3711
                 95% CI : (0.3178, 0.4267)
    No Information Rate : 0.3931
    P-Value [Acc > NIR] : 0.805096

                  Kappa : 0.0833

 Mcnemar's Test P-Value : 0.005472

Statistics by Class:

                     Class: Specific Class: Modern Class: Old_Fashioned Class: Channel
Sensitivity                  0.10909        0.5360               0.4556        0.08333
Specificity                  0.89734        0.5648               0.6974        0.92593
Pos Pred Value               0.18182        0.4437               0.3727        0.16667
Neg Pred Value               0.82807        0.6527               0.7644        0.85034
Prevalence                   0.17296        0.3931               0.2830        0.15094
Detection Rate               0.01887        0.2107               0.1289        0.01258
Detection Prevalence         0.10377        0.4748               0.3459        0.07547
Balanced Accuracy            0.50321        0.5504               0.5765        0.50463
```

```r
# 7
# A naive classifier model is one that does not use any sophistication in order
# to make a prediction, typically making a random or constant prediction.

fct_count(train.wc$weather_source)

# 7. a
110/(110+191+163+86+7)*100 # naive rule
0.4836*100 # naive Bayes

# The accuracy of naive Bayes is 48.36%, the accuracy of naive rule is 19.75%.
# Naive Bayes is much more accurate than naive rule in my model.

# 7. b
# I do not think we should worry too much about the accuracy of less than 50%,
# we are only comparing one variable to the whole data set. If it is than 50%,
# then it means the variable we chose has a small portion of the whole data set.
```

```r
# 8. a
pred.prob <- predict(model, newdata = valid.wc, type="raw")
pred.prob
pred.class <- predict(model, newdata = valid.wc$weather_source)
pred.class

df.lg <- data.frame(actual = valid.wc$weather_source,
                    predict = pred.class, pred.prob)

likely_group <- arrange(df.lg, desc('weather_source')) %>% slice(1:50)
View(likely_group)
str(likely_group)
```

```
> str(likely_group)
'data.frame':   50 obs. of  6 variables:
 $ actual       : Factor w/ 4 levels "Specific","Modern",..: 2 2 4 2 3 2 2 2 1 2 ...
 $ predict      : Factor w/ 4 levels "Specific","Modern",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ Specific     : num  0.0177 0.0773 0.0726 0.1724 0.279 ...
 $ Modern       : num  0.648 0.611 0.645 0.459 0.237 ...
 $ Old_Fashioned: num  0.3216 0.2471 0.0964 0.2402 0.3543 ...
 $ Channel      : num  0.0126 0.0647 0.1864 0.1284 0.1293 ...
```

```r
# 8. b
# By counting the data set, there are 25 actually belonged to this class.

# 8. c
# The likely group has the highest accuracy of 50%, the naive Bayes has the
# second highest accuracy of48.36%, and the naive rule has the lowest accuracy
# of 19.75%.

# 8. d
# For a music app modeler, he may be particularly interested in songs that
# George or a group of people most likely to hear and least likely to hear. Then
# the modeler can predict weather the upcoming album George or that group of
# people may like it.
```

```r
# 9. a
pred.prob <- predict(model, newdata = valid.wc, type = "raw")
pred.class <- predict(model, newdata = valid.wc)

names(valid.wc)

df.9 <- data.frame(actual = valid.wc$weather_source,
                   predict = pred.class, pred.prob)

df.9[valid.wc$ck_weather == FALSE & valid.wc$weather_source == 'Modern' &
       valid.wc$ck_weather_watch == 'Somewhat likely' &
       valid.wc$age == '18 - 29' & valid.wc$female == FALSE &
       valid.wc$hhold_income == '$0 to $9,999' & valid.wc$region == 'Pacific',]
```

```
> pred.prob <- predict(model, newdata = valid.wc, type = "raw")
> pred.class <- predict(model, newdata = valid.wc)
>
> names(valid.wc)
[1] "ck_weather"       "weather_source"   "ck_weather_watch" "age"              "female"
[6] "hhold_income"     "region"
>
> df.9 <- data.frame(actual = valid.wc$weather_source,
+                    predict = pred.class, pred.prob)
>
> df.9[valid.wc$ck_weather == FALSE & valid.wc$weather_source == 'Modern' &
+        valid.wc$ck_weather_watch == 'Somewhat likely' &
+        valid.wc$age == '18 - 29' & valid.wc$female == FALSE &
+        valid.wc$hhold_income == '$0 to $9,999' & valid.wc$region == 'Pacific',]
[1] actual        predict       Specific      Modern        Old_Fashioned Channel
<0 rows> (or 0-length row.names)
```

```r
# 9. b
predict(model, df.9)

# 9. c
model
```