Assignment3 AD 699

Hugi Reyhandani Munggaran

2023-03-30

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(caret)
```

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

```
library(visualize)
library(readr)
library(FNN)
library(tidyverse)
```

```
## ── Attaching packages
## ───────────────────────────────────────
## tidyverse 1.3.2 ──
```

```
## ✔ tibble  3.1.8     ✔ stringr 1.4.0
## ✔ tidyr   1.2.0     ✔ forcats 0.5.1
## ✔ purrr   0.3.4
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
```

```
library(e1071)
```

- K-Nearest Neighbors

Question 1

a. Bohemian Rhapsody

b. I chose Bohemian Rhapsody by Queen because I believe it is a timeless masterpiece that has captured the hearts of people from all walks of life. Whenever I listen to this song, I'm filled with emotions and awe, and I can't help but wonder what it would be like if Freddie were still with us. Additionally, my girlfriend and I share a mutual love and appreciation for this classic tune, which makes it even more special to us. Its such an Intelligent song!

c. Danceability: 39.2% Energy: 40.2% Loudness: -9.961 dB Speechiness: 5.36% Acousticness: 28.8% Instrumentalness: 0.0% Liveness: 24.3% Tempo: 143.883 bpm Duration: 5 minutes and 54 seconds Valence: 22.8%

```
spot100 <- read_csv('spot100.csv')
```

```
## Rows: 100 Columns: 14
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (2): id, name
## dbl (12): duration, energy, key, loudness, mode, speechiness, acousticness, ...
##
## 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.
```

Question 2
a.

```
# Select Bohemian Rhapsody and stored it as a Dataframe name Bohemian
Bohemian<- spot100[spot100$name == "Bohemian Rhapsody", ]
Bohemian <- Bohemian %>%
  rename(duration_ms = duration)
```

Question 3
a. Based on the structure function output, it appears that variable 'target' is a numerical binary variable that represents whether George, the person who uploaded this dataset, liked the song or not.

```
# Import Spotify Dataset
spotify <- read_csv("spotify.csv")
```

```
## New names:
## Rows: 2017 Columns: 17
## ── Column specification
## ──────────────────────────────────────────────────────── Delimiter: "," chr
## (2): song_title, artist dbl (15): ...1, acousticness, danceability,
## duration_ms, energy, instrumenta...
## 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`
```

```
str(spotify)
```

```
## spec_tbl_df [2,017 × 17] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1            : num [1:2017] 0 1 2 3 4 5 6 7 8 9 ...
##  $ acousticness    : num [1:2017] 0.0102 0.199 0.0344 0.604 0.18 0.00479 0.0145 0.
0202 0.0481 0.00208 ...
##  $ danceability    : num [1:2017] 0.833 0.743 0.838 0.494 0.678 0.804 0.739 0.266
0.603 0.836 ...
##  $ duration_ms     : num [1:2017] 204600 326933 185707 199413 392893 ...
##  $ energy          : num [1:2017] 0.434 0.359 0.412 0.338 0.561 0.56 0.472 0.348
0.944 0.603 ...
##  $ instrumentalness: num [1:2017] 2.19e-02 6.11e-03 2.34e-04 5.10e-01 5.12e-01 0.0
0 7.27e-06 6.64e-01 0.00 0.00 ...
##  $ key             : num [1:2017] 2 1 2 5 5 8 1 10 11 7 ...
##  $ liveness        : num [1:2017] 0.165 0.137 0.159 0.0922 0.439 0.164 0.207 0.16
0.342 0.571 ...
##  $ loudness        : num [1:2017] -8.79 -10.4 -7.15 -15.24 -11.65 ...
##  $ mode            : num [1:2017] 1 1 1 1 0 1 1 0 0 1 ...
##  $ speechiness     : num [1:2017] 0.431 0.0794 0.289 0.0261 0.0694 0.185 0.156 0.0
371 0.347 0.237 ...
##  $ tempo           : num [1:2017] 150.1 160.1 75 86.5 174 ...
##  $ time_signature  : num [1:2017] 4 4 4 4 4 4 4 4 4 4 ...
##  $ valence         : num [1:2017] 0.286 0.588 0.173 0.23 0.904 0.264 0.308 0.393
0.398 0.386 ...
##  $ target          : num [1:2017] 1 1 1 1 1 1 1 1 1 1 ...
##  $ song_title      : chr [1:2017] "Mask Off" "Redbone" "Xanny Family" "Master Of N
one" ...
##  $ artist          : chr [1:2017] "Future" "Childish Gambino" "Future" "Beach Hous
e" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ...1 = col_double(),
##   ..   acousticness = col_double(),
##   ..   danceability = col_double(),
##   ..   duration_ms = col_double(),
##   ..   energy = col_double(),
##   ..   instrumentalness = col_double(),
##   ..   key = col_double(),
##   ..   liveness = col_double(),
##   ..   loudness = col_double(),
##   ..   mode = col_double(),
##   ..   speechiness = col_double(),
##   ..   tempo = col_double(),
##   ..   time_signature = col_double(),
##   ..   valence = col_double(),
##   ..   target = col_double(),
##   ..   song_title = col_character(),
##   ..   artist = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
# Convert target variable to factor
spotify$target <- factor(spotify$target)
```

b. Below shows that the unique values for the target variable are 0 and 1. There are 997 records with the value of 0, which means George disliked the song, and 1020 records with the value of 1, which means

George liked the song.

```
# Creating table to find Unique values in 'target'
table(spotify$target)
```

```
##
##    0    1
##  997 1020
```

Question 4

a. The output of the code below is 0 which indicates that there are no NA values in the spotify dataset.

```
# finding NA
sum(is.na(spotify))
```

```
## [1] 0
```

Question 5

```
# remove variable ...1 or X, key, mode and time_signature
spotify <- select(spotify, -c(...1, key, mode, time_signature))
```

Question 6

Using the caret library, I partitioned the Spotify dataset into training and validation sets with a ratio of 60:40 respectively, using a random seed of 121.

```
# Set same seed from assignment 2
set.seed(121)
# Create training (60%) and validation (40%) sets
trainIndex <- createDataPartition(spotify$target, p = 0.6, list = FALSE)
train <- spotify[trainIndex, ]
valid <- spotify[-trainIndex, ]
```

Question 7

```
# Split data into two subsets based on target variable
liked_songs <- subset(train, target == 1)
disliked_songs <- subset(train, target == 0)

# Perform t-tests for all numeric variables
numeric_cols <- c("acousticness", "danceability", "duration_ms", "energy", "instrumen
talness",
                  "liveness", "loudness", "speechiness", "tempo", "valence")

ttest_results <- data.frame(variable = character(),
                            t_statistic = numeric(),
                            p_value = numeric(),
                            stringsAsFactors = FALSE)

for (col in numeric_cols) {
  ttest <- t.test(liked_songs[,col], disliked_songs[,col])
  ttest_results <- rbind(ttest_results,
                         data.frame(variable = col,
                                    t_statistic = round(ttest$statistic,5),
                                    p_value = round(ttest$p.value,5)))
}

ttest_results
```

```
##            variable t_statistic p_value
## t      acousticness    -4.83275 0.00000
## t1     danceability     6.05916 0.00000
## t2      duration_ms     5.38932 0.00000
## t3           energy     1.78250 0.07495
## t4 instrumentalness     4.13209 0.00004
## t5         liveness     1.02442 0.30584
## t6         loudness    -1.26359 0.20667
## t7      speechiness     5.59957 0.00000
## t8            tempo     1.39073 0.16457
## t9          valence     3.64725 0.00028
```

b. Based on the t-test results, the p-values for the variables ‘energy’, ‘liveness’, ‘loudness’, and ‘tempo’ are greater than 0.05. This indicates that there is not a significant difference between the means of these variables for songs that George liked and songs that he did not like. Therefore, we cannot reject the null hypothesis that there is no difference in means between these two groups for these variables.

This code will remove the ‘energy’, ‘liveness’, ‘loudness’, and ‘tempo’ variables from both the ‘train’ and ‘valid’ datasets.

```
train <- train[, -c(4, 6, 7, 9)]
valid <- valid[, -c(4, 6, 7, 9)]
```

c. It may make sense to remove variables from a k-nn model when those variables values are very similar for both outcome classes because these variables do not contribute significantly to the differentiation between the classes. This can lead to noisy and unreliable predictions, as the model is not effectively capturing the underlying patterns and differences in the data. By removing these variables, the model can focus on the most informative variables and potentially improve its predictive accuracy

Question 8

```
# Normalize numeric columns in train and valid dataframes
num_cols <- c("acousticness", "danceability", "duration_ms", "instrumentalness",
              "speechiness", "valence")

train_norm <- preProcess(train[, num_cols], method = c("center", "scale"))
train_norm_df <- predict(train_norm, train[, num_cols])

valid_norm <- preProcess(valid[, num_cols], method = c("center", "scale"))
valid_norm_df <- predict(valid_norm, valid[, num_cols])
```

Question 9

```
# Preprocess and normalize numeric variables in "bohemian" dataframe
Bohemian_norm <- predict(train_norm, Bohemian[, num_cols])

# Use knn() to predict whether George will like the song
k <- 7
predicted_target <- knn(train_norm_df, Bohemian_norm, train$target, k)
predicted_target
```

```
## [1] 1
## attr(,"nn.index")
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]  352  629  109  132  185  196  978
## attr(,"nn.dist")
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## [1,] 1.758125 1.927386 1.997817 2.114009 2.136058 2.195007 2.237923
## Levels: 1
```

The output of predicted_target is 1, indicating that the model predicted that George will like the song “Bohemian Rhapsody”.

```
# Get the indices of the 7 nearest neighbors
nearest_indices <- attr(predicted_target, "nn.index")[1, ]

# Create a data frame with the song titles, artists, and outcome classes of the 7 nea
rest neighbors
nearest <- train[nearest_indices ,c("song_title", "artist", "target")]
nearest
```

```
## # A tibble: 7 × 3
##   song_title                   artist          target
##   <chr>                        <chr>           <fct>
## 1 Wait for the Man             FIDLAR          1
## 2 Zac Brown Band - Hot Country Various Artists 0
## 3 Holdin On                    Flume           1
## 4 It #1                        Ty Segall       1
## 5 Strychnine                   The Sonics      1
## 6 Time-Out                     Terror Train    1
## 7 Waiting                      Dash Berlin     0
```

The 7 nearest neighbors to the “Bohemian Rhapsody” song are:

- “Wait for the Man” by FIDLAR with a target outcome of 1 (George would like it)

- “Zac Brown Band - Hot Country” by Various Artists with a target outcome of 0 (George would not like it)
- “Holdin On” by Flume with a target outcome of 1 (George would like it)
- “It #1” by Ty Segall with a target outcome of 1 (George would like it)
- “Strychnine” by The Sonics with a target outcome of 1 (George would like it)
- “Time-Out” by Terror Train with a target outcome of 1 (George would like it)
- “Waiting” by Dash Berlin with a target outcome of 0 (George would not like it)

Question 10

The highest accuracy is achieved at k=13 with an accuracy of 0.7071960.

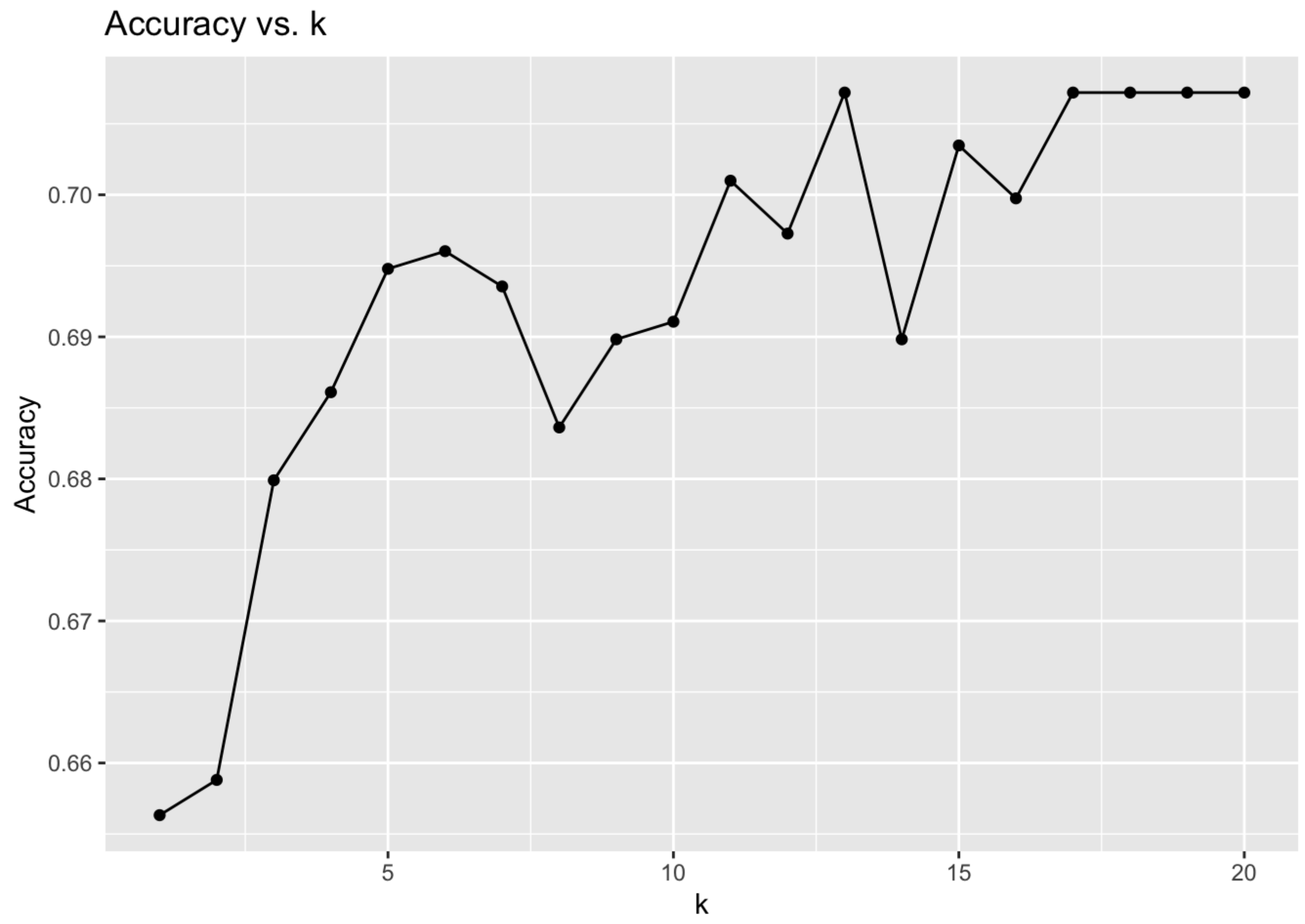
```
# Initialize a data frame with two columns: k, and accuracy
accuracy.df <- data.frame(k = seq(1, 20, 1), accuracy = rep(0, 20))

# Compute knn for different k on validation data
for (i in 1:20) {
  knn.pred <- knn(train_norm_df[, 1:6], valid_norm_df[, 1:6], cl = train$target,
                  k = i)
  accuracy.df[i, 2] <- confusionMatrix(knn.pred, valid$target)$overall[1]
}
print(accuracy.df)
```

```
##     k  accuracy
## 1   1 0.6563275
## 2   2 0.6588089
## 3   3 0.6799007
## 4   4 0.6861042
## 5   5 0.6947891
## 6   6 0.6960298
## 7   7 0.6935484
## 8   8 0.6836228
## 9   9 0.6898263
## 10 10 0.6910670
## 11 11 0.7009926
## 12 12 0.6972705
## 13 13 0.7071960
## 14 14 0.6898263
## 15 15 0.7034739
## 16 16 0.6997519
## 17 17 0.7071960
## 18 18 0.7071960
## 19 19 0.7071960
## 20 20 0.7071960
```

Question 11

```
# Create a ggplot object
ggplot(accuracy.df, aes(x = k, y = accuracy)) + geom_point() + geom_line() + xlab
("k") + ylab("Accuracy") +
  ggtitle("Accuracy vs. k")
```



Question 12 Using this optimal k-value, the algorithm was used to predict whether George would like the song “Bohemian Rhapsody”. The result was the same as before, with a predicted outcome of 1, meaning that George would like the song. This is not surprising given that the accuracy difference between the k-values of 7 and 13 was relatively small (0.6935484 and 0.7071960 respectively).

```
# Run knn with k=13
k <- 13
predicted_target2 <- knn(train_norm_df, Bohemian_norm, train$target, k)
predicted_target2
```

```
## [1] 1
## attr(,"nn.index")
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]  352  629  109  132  185  196  978  807   13   583   351   866  1130
## attr(,"nn.dist")
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
## [1,] 1.758125 1.927386 1.997817 2.114009 2.136058 2.195007 2.237923 2.243677
##          [,9]    [,10]    [,11]    [,12]    [,13]
## [1,] 2.250597 2.262296 2.276294 2.310214 2.316539
## Levels: 1
```

```
# Get the indices of the 13 nearest neighbors
nearest_indices2 <- attr(predicted_target2, "nn.index")[1, ]

# Create a data frame with the song titles, artists, and outcome classes of the 13 ne
arest neighbors
nearest2 <- train[nearest_indices2 ,c("song_title", "artist", "target")]
nearest2
```

```
## # A tibble: 13 × 3
##    song_title                    artist           target
##    <chr>                         <chr>            <fct>
##  1 Wait for the Man              FIDLAR           1
##  2 Zac Brown Band - Hot Country  Various Artists  0
##  3 Holdin On                     Flume            1
##  4 It #1                         Ty Segall        1
##  5 Strychnine                    The Sonics       1
##  6 Time-Out                      Terror Train     1
##  7 Waiting                       Dash Berlin      0
##  8 Hear Me Out                   Risking It All   0
##  9 Girlfriend                    Ty Segall        1
## 10 Gonna Die                     Autre Ne Veut    1
## 11 Max Can't Surf                FIDLAR           1
## 12 Rose Horizon                  Yellow Claw      0
## 13 I Keep Ticking On             The Harmaleighs  0
```

The 13 nearest neighbors to the "Bohemian Rhapsody" song are:

- "Wait for the Man" by FIDLAR with a target outcome of 1 (George would like it)
- "Zac Brown Band - Hot Country" by Various Artists with a target outcome of 0 (George would not like it)
- "Holdin On" by Flume with a target outcome of 1 (George would like it)
- "It #1" by Ty Segall with a target outcome of 1 (George would like it)
- "Strychnine" by The Sonics with a target outcome of 1 (George would like it)
- "Time-Out" by Terror Train with a target outcome of 1 (George would like it)
- "Waiting" by Dash Berlin with a target outcome of 0 (George would not like it)
- "Hear Me Out" by Risking It All with a target outcome of 0 (George would not like it)
- "Girlfriend" by Ty Segall with a target outcome of 1 (George would like it)
- "Gonna Die" by Autre Ne Veut with a target outcome of 1 (George would like it)
- "Max Can't Surf" by FIDLAR with a target outcome of 1 (George would like it)
- "Rose Horizon by" Yellow Claw with a target outcome of 0 (George would not like it)
- "I Keep Ticking On" by The Harmaleighs with a target outcome of 0 (George would not like it)

- Naive Bayes

Question 1

```
library(carData)
Chile <- carData::Chile
```

Question 2

a. The table shows that the variables "age", "education", "income", "statusquo", and "vote" have missing values in 1, 11, 98, 17, and 168 rows, respectively.

```
summary(Chile)
```

```
##  region     population     sex           age        education
##  C :600   Min.   :  3750   F:1379   Min.   :18.00   P   :1107
##  M :100   1st Qu.: 25000   M:1321   1st Qu.:26.00   PS  : 462
##  N :322   Median :175000            Median :36.00   S   :1120
##  S :718   Mean   :152222            Mean   :38.55   NA's:  11
##  SA:960   3rd Qu.:250000            3rd Qu.:49.00
##           Max.   :250000            Max.   :70.00
##                                     NA's   :1
##      income         statusquo         vote
##  Min.   :  2500   Min.   :-1.80301   A   :187
##  1st Qu.:  7500   1st Qu.:-1.00223   N   :889
##  Median : 15000   Median :-0.04558   U   :588
##  Mean   : 33876   Mean   : 0.00000   Y   :868
##  3rd Qu.: 35000   3rd Qu.: 0.96857   NA's:168
##  Max.   :200000   Max.   : 2.04859
##  NA's   :98       NA's   :17
```

```
# Create a subset of the Chile dataset with selected variables
chile_subset <- Chile[, c("region", "population", "sex", "age", "education", "incom
e", "statusquo", "vote")]

# Generate a table of missingness by column
colSums(is.na(chile_subset))
```

```
##     region population        sex        age  education     income  statusquo
##          0          0          0          1         11         98         17
##       vote
##        168
```

i.

```
# subset the dataset to include only complete cases for the "vote" variable
Chile<- Chile[complete.cases(Chile$vote),]

# check no missing values for the "vote" variable
sum(is.na(Chile$vote))
```

```
## [1] 0
```

1. Imputing values for the response variable when preparing data for the modeling process can be problematic as it may introduce bias and lead to overfitting. Bias arises when imputed values do not accurately represent the true underlying relationship between independent variables and the response variable, which can distort the data representation and affect model performance. Overfitting occurs when the model learns to rely on artificially generated data points rather than the true underlying patterns in the data, resulting in poor to perceiving new, unseen data.

ii. In the Chile dataset, the variables “age”, “education”, and “statusquo” have missingness percentages of less than 1%. Thus. in this case, the row containing missing value will be removed

```
# calculate the percentage of missing values for each variable
NApct <- colMeans(is.na(Chile)) * 100
NApct
```

```
##     region population        sex        age  education     income  statusquo 
## 0.00000000 0.00000000 0.00000000 0.03949447 0.39494471 3.27804107 0.51342812 
##       vote 
## 0.00000000
```

```
# Subset the dataset to include only complete cases for "age", "education", and "stat
usquo"
Chile <- Chile[complete.cases(Chile$age, Chile$education, Chile$statusquo),]

# Confirm that there are no missing values for "age", "education", and "statusquo"
colSums(is.na(Chile))
```

```
##     region population        sex        age  education     income  statusquo 
##          0          0          0          0          0         77          0 
##       vote 
##          0
```

iii. new variable called “chile_income” being created. Its includes the values of the “income” variable in the Chile dataset, with the missing values replaced by the text value “Unknown”

```
# replace missing values in the "income" variable with a separate text value
ChileIncome<- ifelse(is.na(Chile$income), "Unknown", Chile$income)

# check no missing values in the ChileIncome
sum(is.na(ChileIncome))
```

```
## [1] 0
```

1. In the Chile dataset, it’s possible that the “NA” values in the “income” variable may represent individuals who choose not to disclose their income or financial information. This may be because of various reasons such as thoughts about financial privacy, distrust of authorities, or personal reasons. Therefore, the “NA” values in the “income” variable may actually be an interesting value as they could provide insights into the attitudes and behaviors related to financial disclosure among the study population.

Question 3

There are no variables currently stored as Character

```
str(Chile)
```

```
## 'data.frame':    2508 obs. of  8 variables:
##  $ region    : Factor w/ 5 levels "C","M","N","S",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ population: int  175000 175000 175000 175000 175000 175000 175000 175000 175000
175000 ...
##  $ sex       : Factor w/ 2 levels "F","M": 2 2 1 1 1 1 2 1 1 2 ...
##  $ age       : int  65 29 38 49 23 28 26 24 41 41 ...
##  $ education : Factor w/ 3 levels "P","PS","S": 1 2 1 1 3 1 2 3 1 1 ...
##  $ income    : int  35000 7500 15000 35000 35000 7500 35000 15000 15000 15000 ...
##  $ statusquo : num  1.01 -1.3 1.23 -1.03 -1.1 ...
##  $ vote      : Factor w/ 4 levels "A","N","U","Y": 4 2 4 2 2 2 2 2 3 2 ...
```

Question 4

The process of binning numerical variables has been completed successfully for all variables except the “population” variable. This is because the “population” variable contains some duplicate values, which causes the quantile() function to return non-unique values for the breaks argument in the cut() function. Therefore, an alternative approach needs to be used to bin this variable to avoid this issue.

```
Chile$age <- cut(Chile$age, breaks = quantile(Chile$age, probs = seq(0, 1, 0.25)), la
bels = c("Young", "Middle-aged", "Older", "Elderly"))

Chile$income <- cut(Chile$income, breaks = quantile(Chile$income, probs = seq(0, 1,
0.25), na.rm = TRUE), labels = c("Low", "Medium", "High", "Very high"))

Chile$statusquo <- cut(Chile$statusquo, breaks = quantile(Chile$statusquo, probs = se
q(0, 1, 0.25)), labels = c("Conservative", "Moderate", "Liberal", "Very liberal"))

# Bin the "population" variable into 4 equal frequency bins using non-quantile method
Chile$population <- cut(Chile$population, breaks = 4, labels = c("Small", "Medium",
"Large", "Very large"))
```

a.

```
# View the frequency distribution of the numerical variable bins using the table() fu
nction
table(Chile$age)
```

```
##
##       Young Middle-aged       Older     Elderly
##         552         650         618         601
```

```
table(Chile$income)
```

```
##
##       Low    Medium      High Very high
##       460       719       697       406
```

```
table(Chile$statusquo)
```

```
##
## Conservative     Moderate      Liberal Very liberal
##          629          624          627          627
```

```
table(Chile$population)
```

```
##
##      Small     Medium      Large Very large
##        869        310        133       1196
```

b. Equal width binning involves dividing the range of the variable into a fixed number of equally sized intervals or bins. On the other hand, Equal frequency binning involves dividing the variable into a fixed number of bins such that each bin contains an equal number of observations.

Equal frequency binning can be preferable for some scenarios such as when using Naive Bayes classifiers because it help to manage skewed data and outliers by distributing an equal number of data points across bins. This approach can lead to a more balanced representation of the data, improving the algorithm’s performance and generalization, especially when the contains like extreme values.

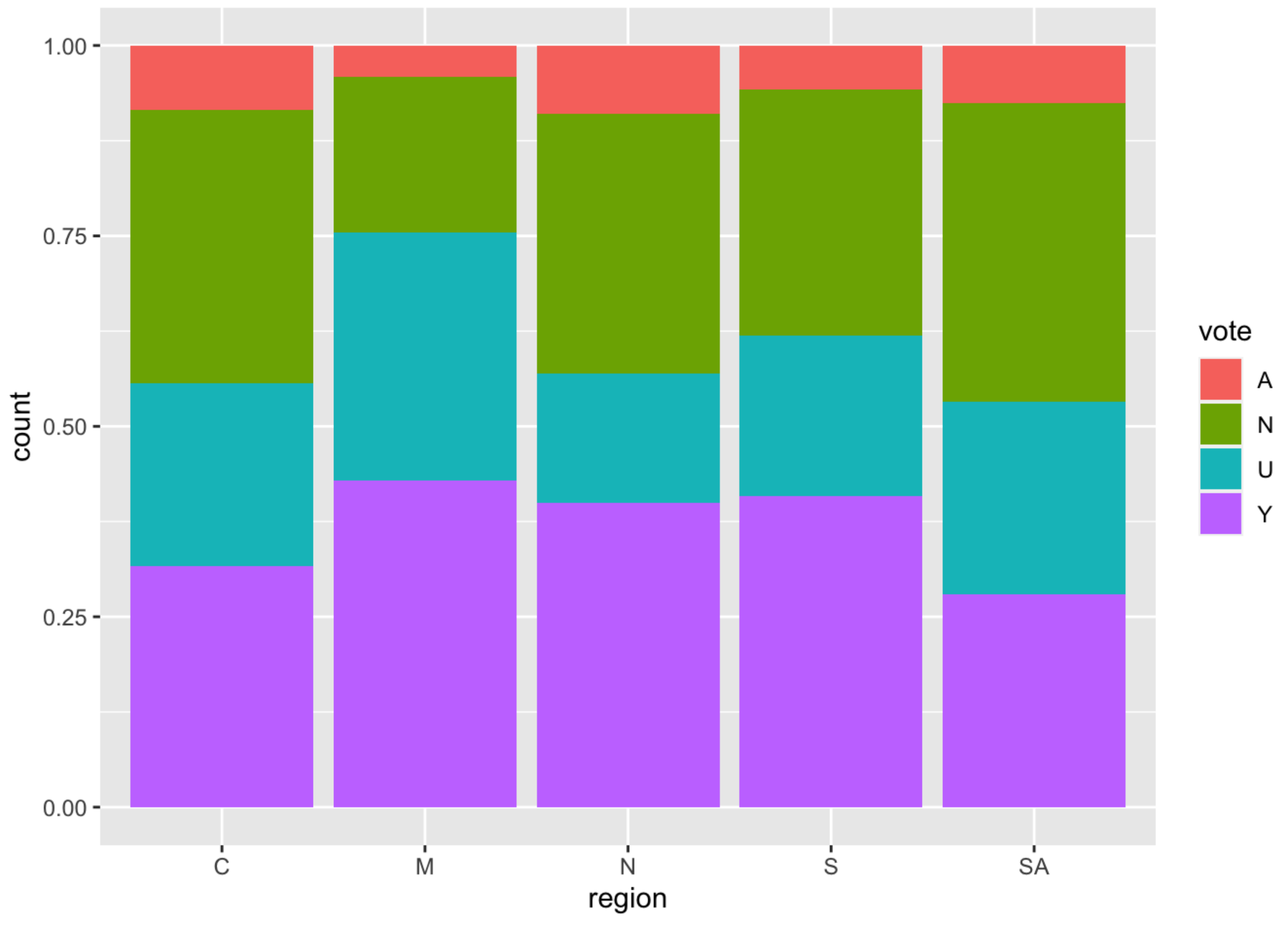
Question 5

```
set.seed(121)
# Create index for training and validation sets
index <- createDataPartition(Chile$vote, p = 0.6, list = FALSE)
ChileTrain <- Chile[index, ]
ChileValid <- Chile[-index, ]
```

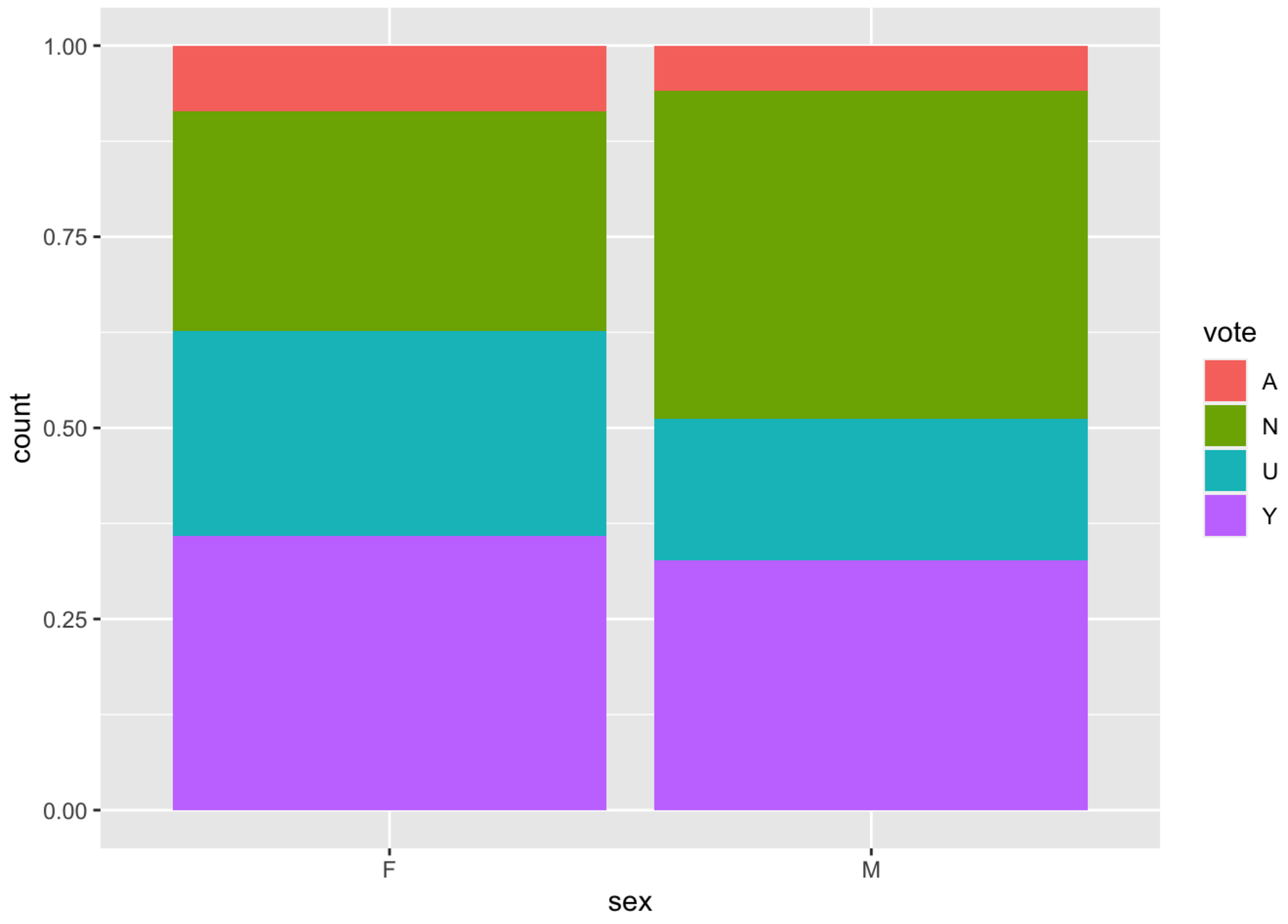
Question 6

Based on the barplots, it appears that the “income” and “sex” variable may not have a strong amount of predictive power in a naive Bayes model as the distribution of the vote across the regions is relatively similar. Therefore, we can drop the “income” and “sex” variable from our dataset.

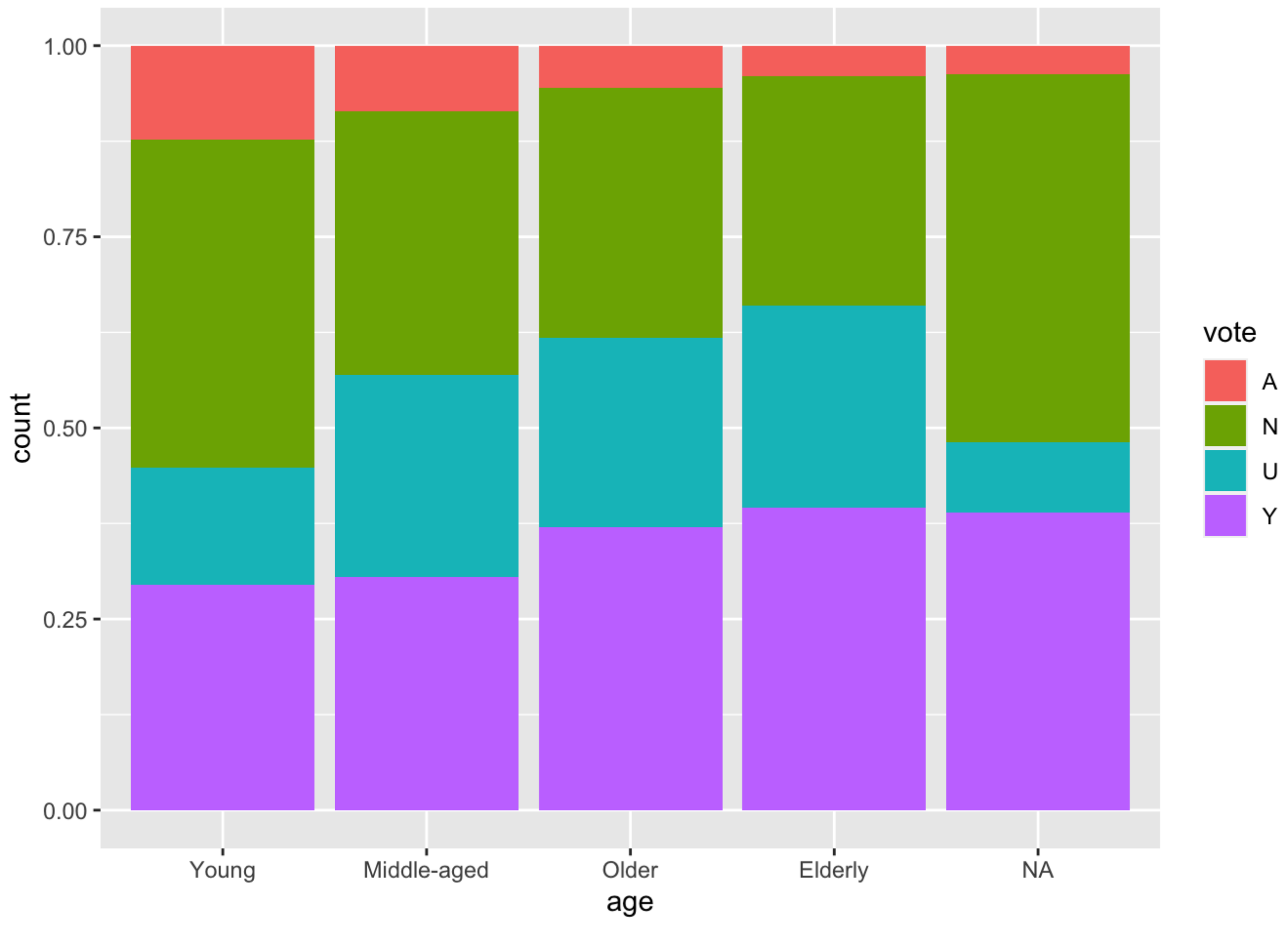
```
# Proportional barplot for region variable
ggplot(ChileTrain, aes(x = region, fill = vote)) +
  geom_bar(position = "fill")
```



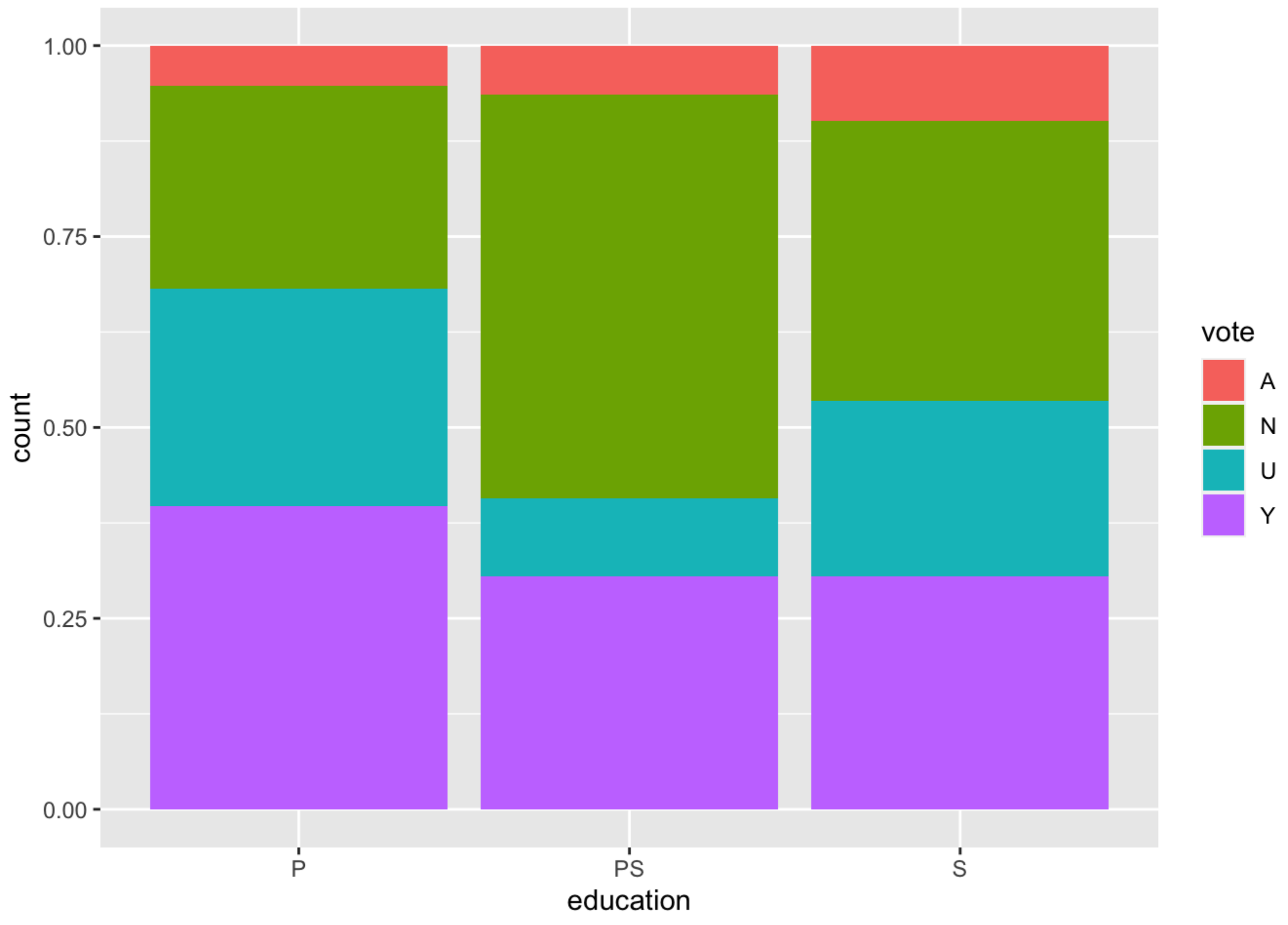
```
# Proportional barplot for sex variable
ggplot(ChileTrain, aes(x = sex, fill = vote)) +
  geom_bar(position = "fill")
```



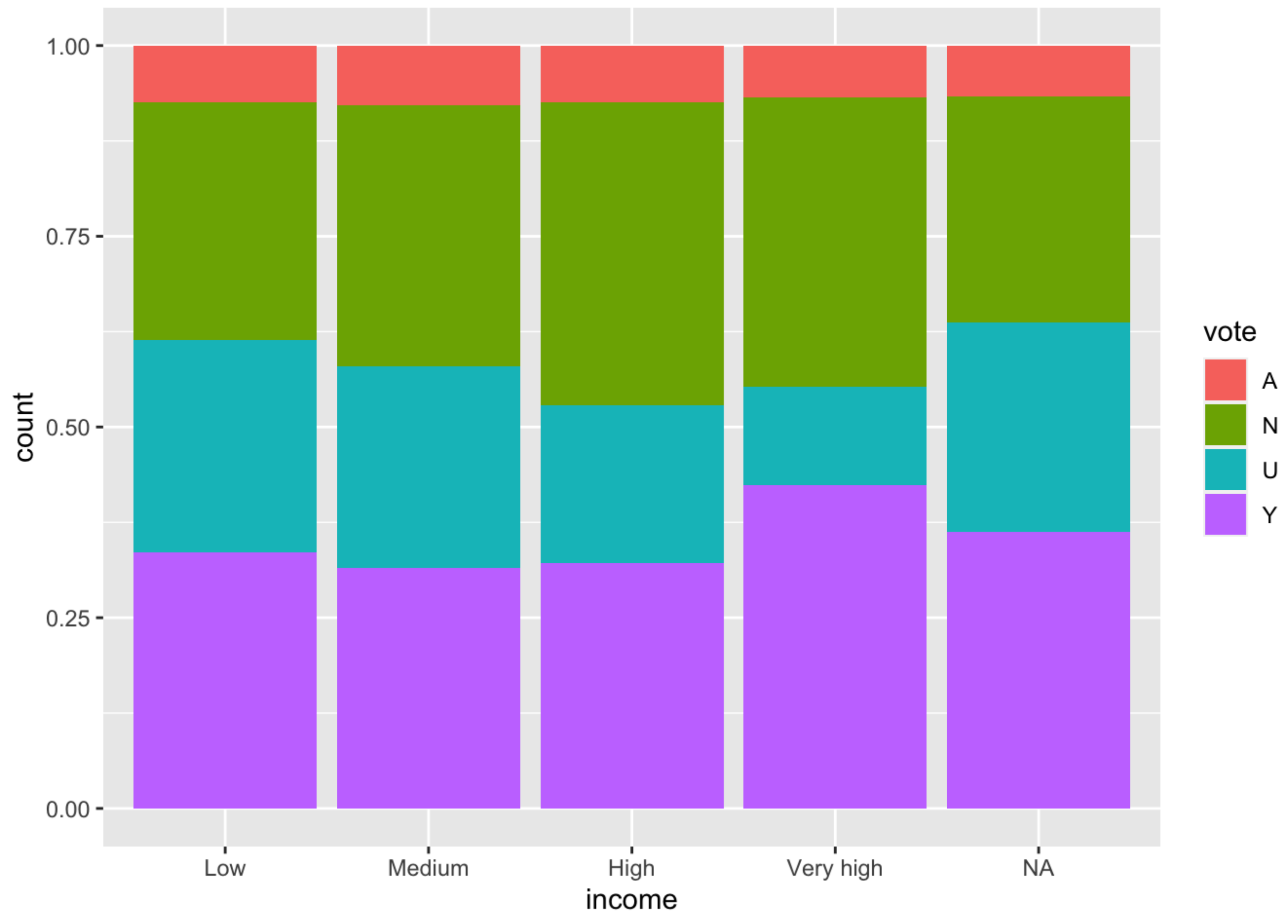
```
# Proportional barplot for age_bin variable
ggplot(ChileTrain, aes(x = age, fill = vote)) +
  geom_bar(position = "fill")
```



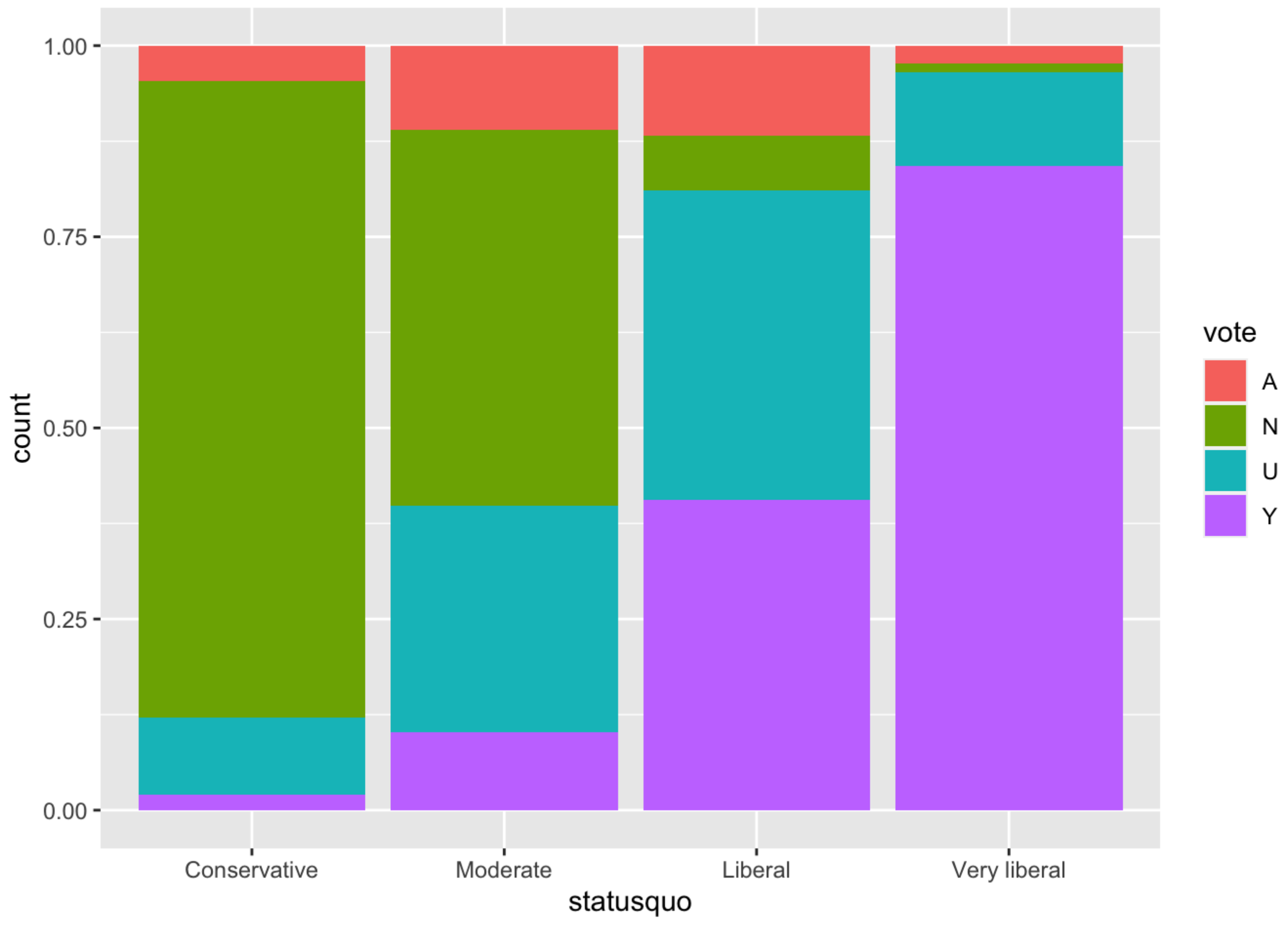
```
# Proportional barplot for education variable
ggplot(ChileTrain, aes(x = education, fill = vote)) +
  geom_bar(position = "fill")
```



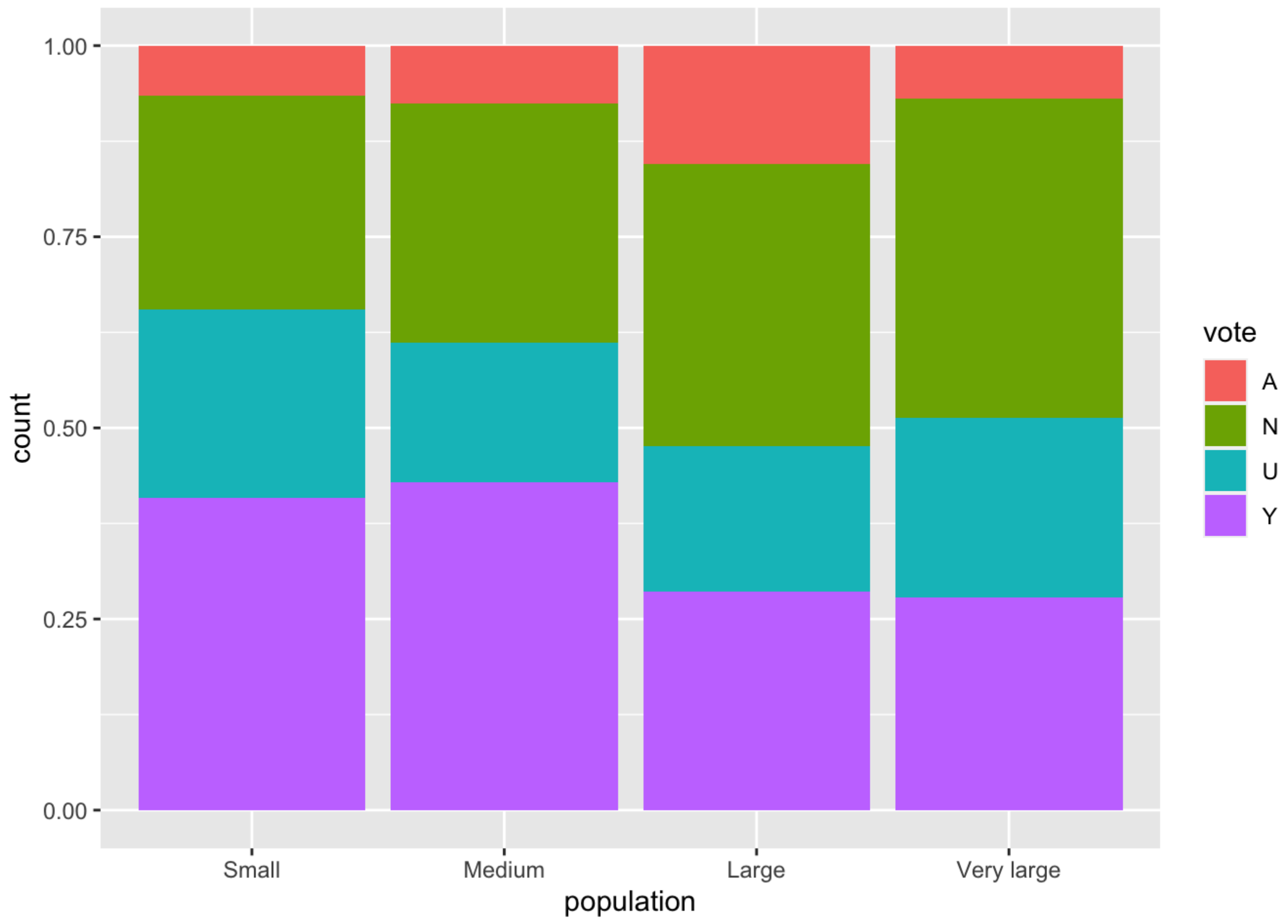
```
# Proportional barplot for income_bin variable
ggplot(ChileTrain, aes(x = income, fill = vote)) +
  geom_bar(position = "fill")
```



```
# Proportional barplot for statusquo_bin variable
ggplot(ChileTrain, aes(x = statusquo, fill = vote)) +
  geom_bar(position = "fill")
```



```
# Proportional barplot for population_bin variable
ggplot(ChileTrain, aes(x = population, fill = vote)) +
  geom_bar(position = "fill")
```



```
# drop income variable
ChileTrain <- ChileTrain[, !names(ChileTrain) %in% c("income","sex")]
ChileValid <- ChileValid[, !names(ChileValid) %in% c("income","sex")]
```

Question 7

```
# Build NB model
ChileNb<- naiveBayes(vote~., data = ChileTrain)
print(ChileNb)
```

```
## 
## Naive Bayes Classifier for Discrete Predictors
## 
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
## 
## A-priori probabilities:
## Y
##          A          N          U          Y 
## 0.07370518 0.35325365 0.22974768 0.34329349 
## 
## Conditional probabilities:
##    region
## Y            C          M          N          S         SA
##   A 0.27027027 0.01801802 0.16216216 0.20720721 0.34234234
##   N 0.23872180 0.01879699 0.12781955 0.24248120 0.37218045
##   U 0.24566474 0.04624277 0.09826590 0.24277457 0.36705202
##   Y 0.21663443 0.04061896 0.15473888 0.31528046 0.27272727
## 
##    population
## Y        Small     Medium      Large Very large
##   A 0.30630631 0.13513514 0.11711712 0.44144144
##   N 0.27443609 0.11654135 0.05827068 0.55075188
##   U 0.37283237 0.10404624 0.04624277 0.47687861
##   Y 0.41199226 0.16441006 0.04642166 0.37717602
## 
##    age
## Y       Young Middle-aged     Older   Elderly
##   A 0.3761468   0.3119266 0.1743119 0.1376147
##   N 0.2826087   0.2707510 0.2233202 0.2233202
##   U 0.1495601   0.3079179 0.2521994 0.2903226
##   Y 0.1975806   0.2439516 0.2580645 0.3004032
## 
##    education
## Y            P         PS          S
##   A 0.29729730 0.15315315 0.54954955
##   N 0.31015038 0.26315789 0.42669173
##   U 0.51156069 0.07803468 0.41040462
##   Y 0.47775629 0.15667311 0.36557060
## 
##    statusquo
## Y   Conservative    Moderate     Liberal Very liberal
##   A  0.162162162 0.360360360 0.396396396  0.081081081
##   N  0.605263158 0.336466165 0.050751880  0.007518797
##   U  0.112716763 0.312138728 0.439306358  0.135838150
##   Y  0.015473888 0.071566731 0.294003868  0.618955513
```

Question 8

After running the confuion matrix it appears that the model's accuracy on the training set was slightly better than on the validation set, with accuracy percentages of 0.661 and 0.638 respectively. Since the difference in accuracy between the two sets is not significant, it suggests that the model is not overfitting to the training data and has a good chance of performing well on new and unseen data. Nonetheless, there is still room for improvement in the model's predictive power.

```
# Predict on training and validation set
trainpred <- predict(ChileNb, ChileTrain, type = "class")
validpred <- predict(ChileNb, ChileValid, type = "class")

# Confusion matrix and accuracy for training set
confusionMatrix(trainpred, ChileTrain$vote)$table
```

```
##           Reference
## Prediction   A   N   U   Y
##          A   4   0   3   2
##          N  53 475 114  32
##          U  20  38 127  93
##          Y  34  19 102 390
```

```
acctrain <- confusionMatrix(trainpred, ChileTrain$vote)$overall[1]

# Confusion matrix and accuracy for validation set
confusionMatrix(validpred, ChileValid$vote)$table
```

```
##           Reference
## Prediction   A   N   U   Y
##          A   1   0   3   5
##          N  42 306  81  24
##          U   9  26  74  57
##          Y  22  22  72 258
```

```
accvalid <- confusionMatrix(validpred, ChileValid$vote)$overall[1]



acctrain
```

```
##  Accuracy
## 0.6613546
```

```
accvalid
```

```
##  Accuracy
## 0.6377246
```

Question 9

The naive rule for classification is an approach where all records are classified based on the most frequent class in the training set.

If we used the naive rule as an approach to classification for our training set, we would classify all the records based on the most frequent class in the training set. Based on the training set's distribution, we would classify all records as "N" (will vote no against Pinochet) since it is the most frequent class in the training set.

a. We can calculate the percentage of the most frequent variable in our training and validation datasets by using the below code. The resulting accuracy of the naive rule on both the training and validation sets is 0.353 or 35.3%.

When we compare this result with the accuracy of our Naive Bayes model (on question 8), it is evident that the model outperforms the naive rule by a significant margin, with an accuracy improvement of around 31% on the training set and 28.5% on the validation set

```
# Create a vector of "N" predictions for the training set
naive_preds <- rep("N", nrow(ChileTrain))

# Calculate the accuracy of the naive rule
naive_accuracy <- sum(naive_preds == ChileTrain$vote) / nrow(ChileTrain)

naive_accuracy
```

```
## [1] 0.3532537
```

Question 10

```
# predict the validation set to be most likely vote yes
pred.probs <- predict(ChileNb, newdata = ChileValid, type = "raw")
df_probs <- data.frame(vote = ChileValid$vote, predyes = pred.probs[, "Y"])

# subset 100 records
subset <- df_probs %>% top_n(100, predyes) %>% slice(1:100)

print(subset)
```

```
##     vote   predyes
## 1      Y 0.8899958
## 2      N 0.8983406
## 3      A 0.8774327
## 4      Y 0.9332531
## 5      Y 0.9198988
## 6      Y 0.8774327
## 7      Y 0.9198988
## 8      Y 0.9250410
## 9      Y 0.9055304
## 10     Y 0.9117835
## 11     Y 0.9250410
## 12     A 0.8730867
## 13     Y 0.8778519
## 14     Y 0.9073637
## 15     Y 0.8730867
## 16     Y 0.8836176
## 17     Y 0.9332531
## 18     U 0.9079359
## 19     Y 0.9116501
## 20     Y 0.9146824
## 21     Y 0.9084636
## 22     Y 0.9079359
## 23     Y 0.8804927
## 24     Y 0.9116501
## 25     Y 0.9079359
## 26     Y 0.9116501
## 27     Y 0.8726585
## 28     A 0.9055304
## 29     Y 0.9293150
## 30     Y 0.9079359
## 31     Y 0.9332531
## 32     Y 0.8987658
## 33     Y 0.9116501
## 34     Y 0.9126649
## 35     Y 0.9126649
## 36     Y 0.9215654
## 37     Y 0.8978826
## 38     Y 0.9140513
## 39     U 0.8666461
## 40     Y 0.9403654
## 41     Y 0.8666461
## 42     Y 0.8985371
## 43     Y 0.8666461
## 44     Y 0.8985371
## 45     U 0.8985371
## 46     Y 0.8776158
## 47     Y 0.9215654
## 48     Y 0.9048289
## 49     A 0.8795115
## 50     Y 0.8666461
## 51     Y 0.8828378
## 52     Y 0.8956779
## 53     Y 0.8985371
## 54     Y 0.8985371
## 55     Y 0.8799189
## 56     Y 0.8828378
## 57     Y 0.8985371
## 58     U 0.8666461
## 59     Y 0.8828378
## 60     Y 0.9361540
## 61     Y 0.8795115
## 62     Y 0.8666461
## 63     Y 0.8956779
## 64     Y 0.8795115
## 65     Y 0.8795115
## 66     Y 0.8956779
## 67     Y 0.8666461
## 68     Y 0.8748097
## 69     Y 0.9145297
## 70     Y 0.9181658
## 71     Y 0.8748653
## 72     Y 0.9048289
## 73     Y 0.9215654
## 74     Y 0.9331574
## 75     U 0.9145297
## 76     Y 0.8795115
## 77     Y 0.8956779
## 78     Y 0.9048289
## 79     Y 0.8666461
## 80     Y 0.8985371
## 81     Y 0.8666461
## 82     U 0.8985371
## 83     U 0.8795115
## 84     Y 0.8828378
## 85     Y 0.9244829
## 86     Y 0.8776158
## 87     Y 0.9287726
## 88     Y 0.8978826
## 89     Y 0.9084013
## 90     Y 0.8985371
## 91     Y 0.8880486
## 92     Y 0.9145297
## 93     U 0.8666461
## 94     U 0.8795115
## 95     N 0.8666461
## 96     Y 0.8963148
## 97     U 0.9215654
## 98     Y 0.9215654
## 99     Y 0.8956779
## 100    Y 0.9403654
```

a. There were 84 people who actually voted “YES” among the 100 records predicted to be most likely to vote “YES” by the model. The accuracy for these predictions is 84%, which is higher than the overall accuracy of the model on the validation set (63.77%). This suggests that the model performs better at predicting the outcome for those who are most likely to vote “YES” than for the general population.

```
# Count the actual number of people who voted "YES"
num_actual_yes <- subset %>% filter(vote == "Y") %>% nrow()
num_actual_yes
```

```
## [1] 84
```

```
# Calculate the accuracy
accuracy <- num_actual_yes / nrow(subset)
accuracy
```

```
## [1] 0.84
```

b. Identifying a subset of individuals who are most likely to vote “YES” can be valuable information for a political party. For example, the party could use this information to focus their resources on these individuals in order to encourage them to persuade another individual that have not been vote and lead them to vote “YES”. The party could tailor their messaging to these individuals based on their characteristics and the factors that are most strongly associated with a “YES” vote. By focusing their efforts on those who are most likely to vote “YES”, the party could potentially increase their chances of success in the election.

Question 11 a. For this question, the record chosen was from row number 8 in the ‘ChileTrain’ dataset. The individual is from the northern city of Santiago and is classified as a conservative. According to their voting intention, they will vote against Pinochet in the referendum.

```
row_8 <- ChileTrain[7,]
row_8
```

```
##   region population   age education    statusquo vote
## 8      N      Large Young         S Conservative    N
```

b.The model predicted that the person from row 8 in the training set voted “N” (against Pinochet)

```
row8 <- predict(ChileNb, ChileTrain[7,], type="class")
row8
```

```
## [1] N
## Levels: A N U Y
```

c. The value 0.01107169 is the predicted probability that the person from row 8 in the training set would vote “Yes” (will vote Pinochet) according to the Naive Bayes model.

```
pred <- predict(ChileNb, row_8, type = "raw")
prob_Y <- pred[1, "Y"]
prob_Y
```

```
##          Y
## 0.01107169
```

d. To calculate the probability of row 8 voting “yes” using the A-priori probabilities and Conditional probabilities from the ChileNB model (result in Question 7), we need to compute the probability of row 8 voting “yes” given the conditional probabilities for all the predictors in row 8. This will be the numerator. We also need to calculate the probability of row 8 voting “no”, “abstain”, and “undecided” given the same set of conditional probabilities for each of these outcomes. These probabilities will form the denominators. Finally, we can calculate the probability of row 8 voting “yes” by dividing the numerator by the sum of the denominators and the numerator.

```
numerator <- 0.34329349*0.15473888*0.04642166*0.1975806*0.36557060*0.015473888
denom1 <- 0.07370518*0.16216216*0.11711712*0.3761468*0.54954955*0.162162162
denom2 <- 0.35325365*0.12781955*0.05827068*0.2826087*0.42669173*0.605263158
denom3 <- 0.22974768*0.09826590*0.04624277*0.1495601*0.41040462*0.112716763

numerator/(numerator+denom1+denom2+denom3)
```

```
## [1] 0.01107168
```