R Project: Spotify Skip Data

Tessa Mitchell

Data Cleaning

The Spotify data used was downloaded from https://www.aicrowd.com/challenges/spotify-sequential-skip-prediction-challenge/dataset_files. It is the first file in the Training_Set_Split_Download.txt. The dataset was published by Spotify as part of a challenge to predict whether songs will be skipped or not. In this report, I will be taking on Spotify's challenge and trying to predict whether or not a song will be skipped.

```
df1 <- read.csv("spotifyDataSkip1.csv")
nrow(df1)
```

Since there are 2,990,609 rows in this csv file, my computer runs out of RAM while trying to process the data, despite having 16 gigs of RAM. This is a common issue with R that I and many others have run in to before. Therefore, I will take a subset of this data as shown below.

```
set.seed(1234)
i <- sample(1:nrow(df1), 100000, replace = FALSE)
df2 <- df1[i, 2:21]
# takes the first 100000 rows of the data
# removes column 1: sesssion_id since it is unnecessary
write.csv(df2, "spotifyDataSkip2.csv")
```

In order to properly knit the file, I wrote the sample dataset into a csv file and then read that csv file in knitting. The above code chunks were run initially but were not run in the knitting process. All the code chunks below were run in knitting.

```
df2 <- read.csv("spotifyDataSkip2.csv")
```

```
names(df2)
```

```
##  [1] "X"                                "session_position"
##  [3] "session_length"                   "track_id_clean"
##  [5] "skip_1"                           "skip_2"
##  [7] "skip_3"                           "not_skipped"
##  [9] "context_switch"                   "no_pause_before_play"
## [11] "short_pause_before_play"          "long_pause_before_play"
## [13] "hist_user_behavior_n_seekfwd"     "hist_user_behavior_n_seekback"
## [15] "hist_user_behavior_is_shuffle"    "hour_of_day"
## [17] "date"                             "premium"
## [19] "context_type"                     "hist_user_behavior_reason_start"
## [21] "hist_user_behavior_reason_end"
```

Thes are the attributes in this dataset, which are described in the Data Descriptions file at https://www.aicrowd.com/challenges/spotify-sequential-skip-prediction-challenge/dataset_files.

Now I will clean this subset of the data. First, I want to ensure that ever attribute was processed correctly by the read.csv() function. This is done by using typeof() in the console to check the type of the variable of each attribute column, and then correcting the ones that need to be corrected. I also remove the first column, which is an arbitrary attribute created by the process of writing and re-reading the data from ,csv files.

```
df3 <- df2[,2:21] # creates a copy of data to clean and removes first row

# converts character "true" and "false" to boolean
df3$skip_1 <- df2$skip_1 == "true"
df3$skip_2 <- df2$skip_2 == "true"
df3$skip_3 <- df2$skip_3 == "true"
df3$not_skipped <- df2$not_skipped == "true"
df3$hist_user_behavior_is_shuffle <- df2$hist_user_behavior_is_shuffle == "true"

# converts integers 0 and 1 to boolean
df3$context_switch <- df2$context_switch == 1
df3$no_pause_before_play <- df2$no_pause_before_play == 1
df3$short_pause_before_play <- df2$short_pause_before_play == 1
df3$long_pause_before_play <- df2$long_pause_before_play == 1

# converts characters to factors
df3$context_type <- factor(df2$context_type)
df3$hist_user_behavior_reason_start <-
  factor(df2$hist_user_behavior_reason_start)
df3$hist_user_behavior_reason_end <- factor(df2$hist_user_behavior_reason_end)
df3$premium <- factor(df3$premium)

# converts characters to date
df3$date <- as.Date(df3$date, "%Y-%m-%d")
```

This data set includes 3 variables - skip__1, skip__2, skip__3 - which indicate at which part of the song the user chose to skip it (if it was skipped at all). It also includes a variable not__skipped, which is fairly straightforward. In order to simplify this project, I have decided to simply predict whether a song was skipped at all and ignored when it was skipped. Therefore, I created this variables skipped, which is the negation of not__skipped, and removed the other variables (skip__1, skip__2, skip__3, and not__skipped). I also removed the track__id__clean variable. Since we have no other information about the actual song being skipped except track__id__clean, which is simply a randomly assigned id number for the song, I have decided to remove this attribute as well.

```
df3$skipped <- ifelse(df3$not_skipped == FALSE, 1L, 0L)
df4 <- cbind(df3[,1:2], df3[,8:21])
```

Data Exploration

What does the data look like?

```
str(df4)
```

```
## 'data.frame':    100000 obs. of  16 variables:
##  $ session_position               : int  15 7 10 16 9 3 19 7 7 12 ...
##  $ session_length                 : int  18 15 16 20 20 14 20 19 14 20 ...
##  $ context_switch                 : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ no_pause_before_play           : logi  TRUE TRUE FALSE TRUE TRUE TRUE ...
##  $ short_pause_before_play        : logi  FALSE FALSE TRUE FALSE FALSE FALSE ...
##  $ long_pause_before_play         : logi  FALSE FALSE TRUE FALSE FALSE FALSE ...
##  $ hist_user_behavior_n_seekfwd   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hist_user_behavior_n_seekback  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hist_user_behavior_is_shuffle  : logi  TRUE FALSE FALSE FALSE FALSE FALSE ...
```

```
##  $ hist_user_behavior_n_seekback : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ hour_of_day                     : int  17 13 19 4 18 19 18 14 12 22 ...
##  $ date                            : Date, format: "2018-07-15" "2018-07-15" ...
##  $ premium                         : Factor w/ 2 levels "false","true": 2 2 2 2 1 1 2 1 2 2 ...
##  $ context_type                    : Factor w/ 6 levels "catalog","charts",..: 1 1 3 3 3 6 3 6 6 5 ..
##  $ hist_user_behavior_reason_start: Factor w/ 9 levels "appload","backbtn",..: 5 8 8 5 5 5 2 5 3 5 .
##  $ hist_user_behavior_reason_end  : Factor w/ 7 levels "backbtn","clickrow",..: 4 7 4 4 4 4 7 4 4 4
##  $ skipped                         : int  1 0 1 1 1 1 0 1 1 1 ...
```

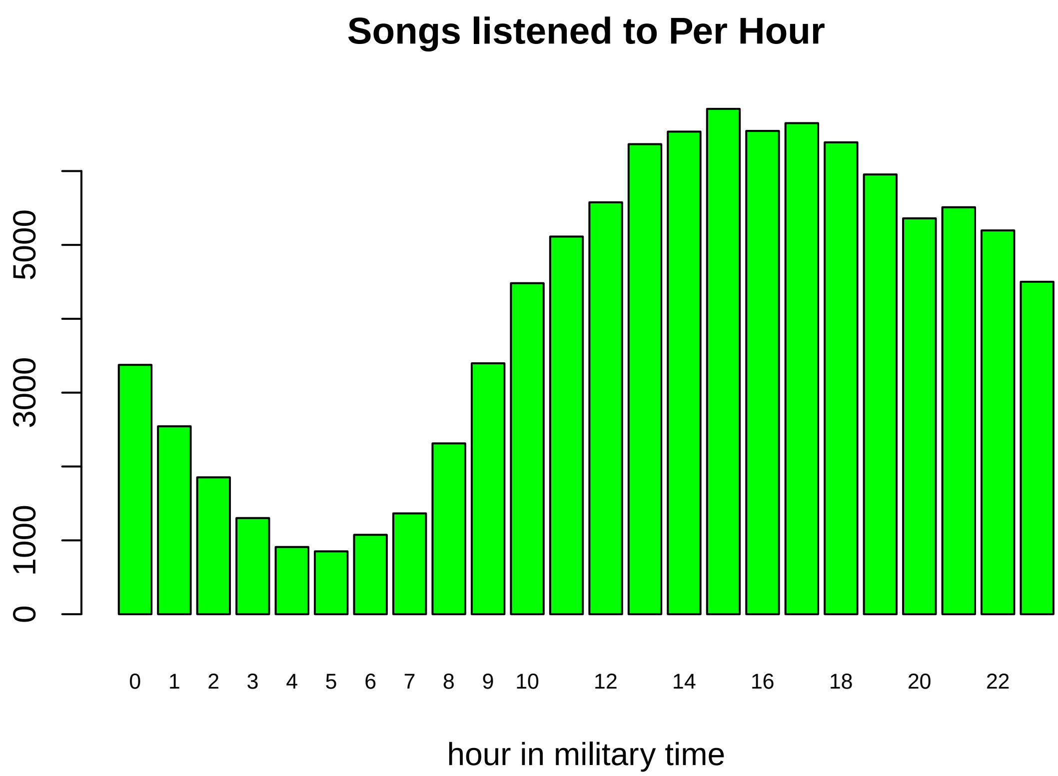
We have integer, logical, factor, and date data types. Since date data types cannot be easily used in machine learning algorithms, we will mostly focus on the other variables.

```r
summary(df4)
```

```
##  session_position session_length  context_switch  no_pause_before_play
##  Min.   : 1.000   Min.   :10.00   Mode :logical   Mode :logical
##  1st Qu.: 5.000   1st Qu.:15.00   FALSE:95749     FALSE:23423
##  Median : 9.000   Median :20.00   TRUE :4251      TRUE :76577
##  Mean   : 9.283   Mean   :17.61
##  3rd Qu.:14.000   3rd Qu.:20.00
##  Max.   :20.000   Max.   :20.00
##
##  short_pause_before_play long_pause_before_play hist_user_behavior_n_seekfwd
##  Mode :logical           Mode :logical          Min.   : 0.00000
##  FALSE:85060             FALSE:82445            1st Qu.: 0.00000
##  TRUE :14940             TRUE :17555            Median : 0.00000
##                                                 Mean   : 0.03748
##                                                 3rd Qu.: 0.00000
##                                                 Max.   :39.00000
##
##  hist_user_behavior_n_seekback hist_user_behavior_is_shuffle  hour_of_day
##  Min.   :  0.00000           Mode :logical                  Min.   : 0.00
##  1st Qu.:  0.00000           FALSE:69634                    1st Qu.:11.00
##  Median :  0.00000           TRUE :30366                    Median :15.00
##  Mean   :  0.04374                                          Mean   :14.15
##  3rd Qu.:  0.00000                                          3rd Qu.:19.00
##  Max.   :151.00000                                          Max.   :23.00
##
##       date              premium                     context_type
##  Min.   :2017-03-02   false:19185   catalog              :22655
##  1st Qu.:2018-07-14   true :80815   charts               : 1405
##  Median :2018-07-15                 editorial_playlist   :21104
##  Mean   :2018-07-14                 personalized_playlist: 1698
##  3rd Qu.:2018-07-15                 radio                :13070
##  Max.   :2018-08-22                 user_collection      :40068
##
##  hist_user_behavior_reason_start hist_user_behavior_reason_end    skipped
##  fwdbtn   :47270                 backbtn  : 7537               Min.   :0.0000
##  trackdone:33383                 clickrow :    3               1st Qu.:0.0000
##  clickrow :10341                 endplay  : 8845               Median :1.0000
##  backbtn  : 7568                 fwdbtn   :48152               Mean   :0.6613
##  appload  : 1186                 logout   :  297               3rd Qu.:1.0000
##  playbtn  :   94                 remote   :  166               Max.   :1.0000
##  (Other)  :  158                 trackdone:35000
```

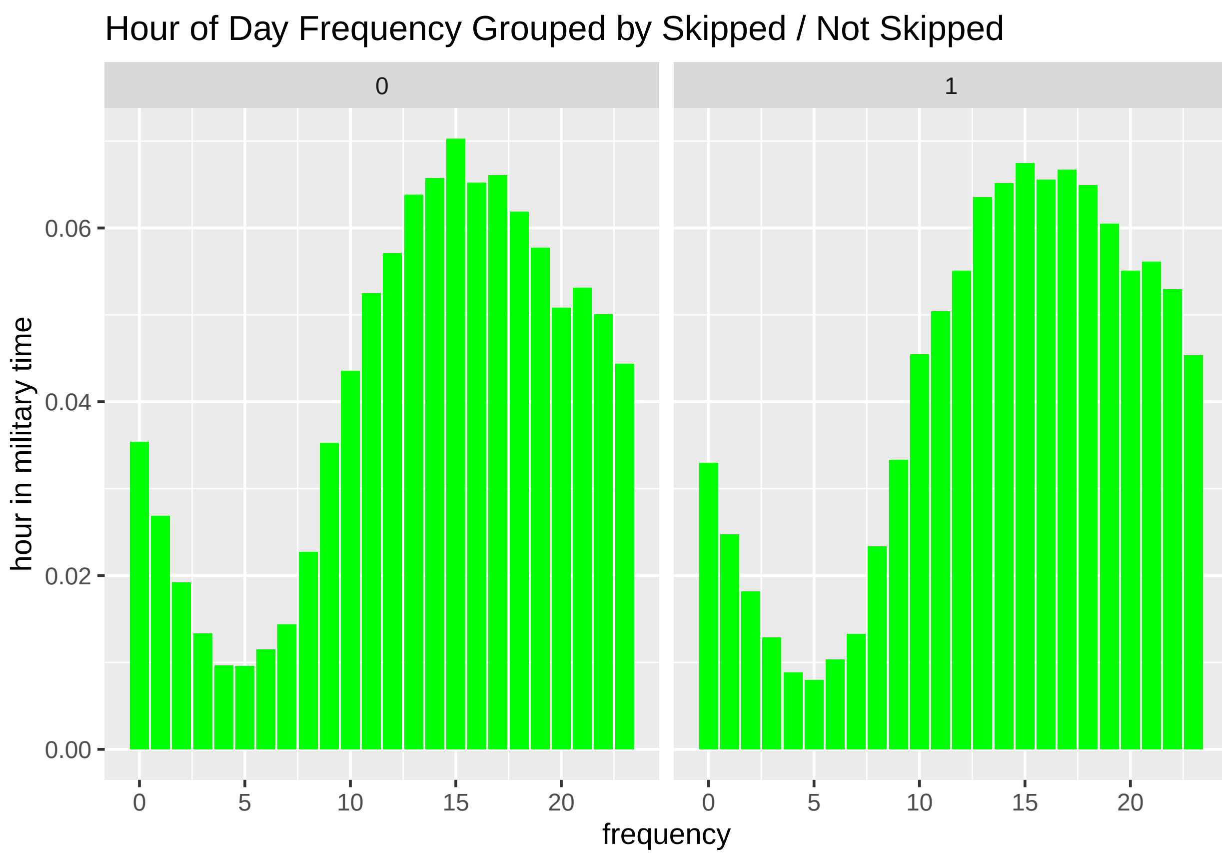
What hours of the day are people most likely to listen to Spotify? What hours of the day are people most likely to skip songs?

```
table1 <- table(df4$hour_of_day)
barplot(table1, col = c("green"), cex.names = 0.7, main = "Songs listened to Per Hour",
        xlab = "hour in military time")
```



We can see that people use Spotify the most in the 14th or 15th hour, which is around 3:00pm. People use Spotify the least during the 4th and 5th hour, or around 5:00am.

```
library(ggplot2)
ggplot(df4) +
  geom_bar(aes(x = hour_of_day, y = ..prop.., group = skipped),
           fill = "green") + facet_wrap(~ skipped) +
  labs(title = "Hour of Day Frequency Grouped by Skipped / Not Skipped") +
  xlab("frequency") + ylab("hour in military time")
```



Here we see the hour of day that users listen to Spotify, separated by the songs they listen through fully (left) and the songs they skip (right). Both graphs are about the same, suggesting that the time of day does not heavily influence whether or not a user skips a song.

What dates does our data span?

```
range(df4$date)
```

```
## [1] "2017-03-02" "2018-08-22"
```

Our data spans only a couple of months, from May 18th to July 16th of 2018.

What is the average length of a Spotify listening session?

```
unique_sessions <- df4[df4$session_length == df4$session_position, ]
    # filters out unique sessions by considering
    # only the last song of each session
mean(unique_sessions$session_length)
```

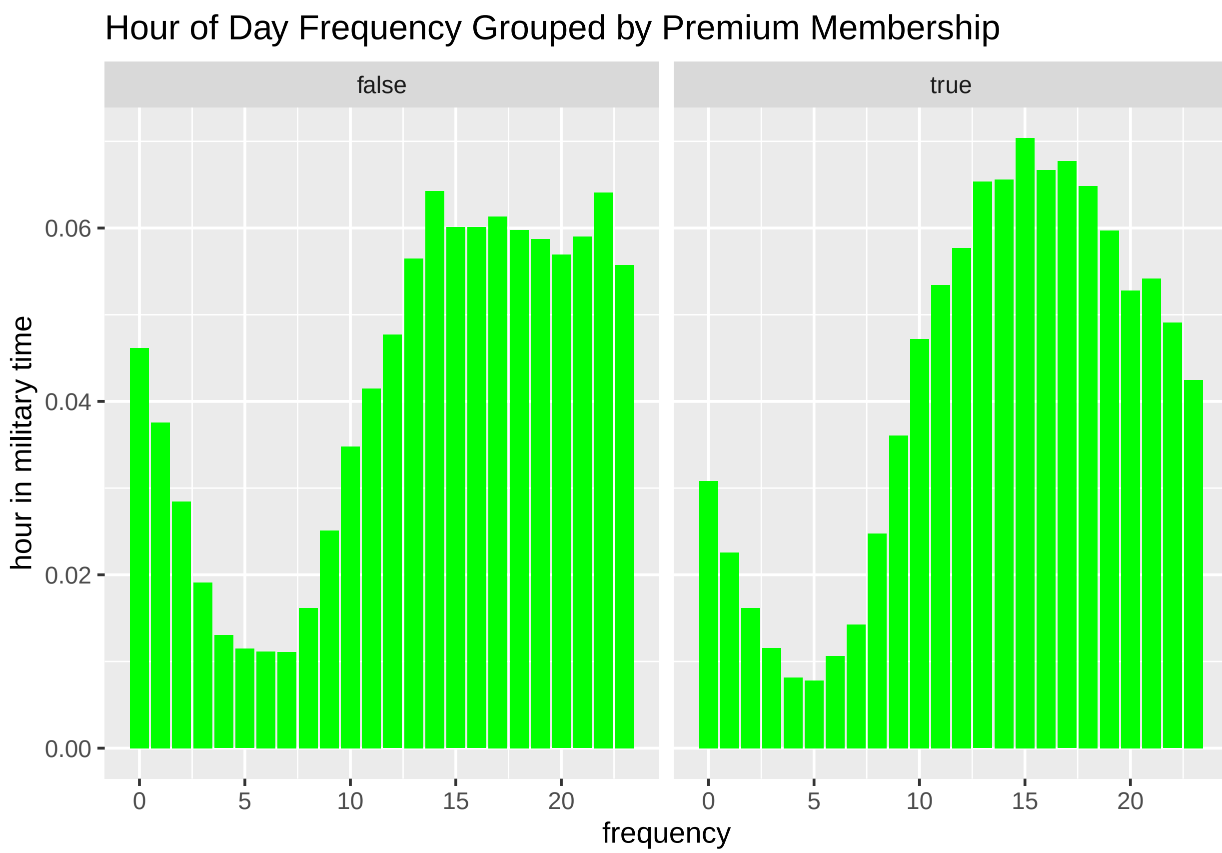
```
## [1] 16.66615
```

When a person listens to Spotify, they listen to (on average) 17 songs, including skipped songs.

How does having premium change a user's habits?

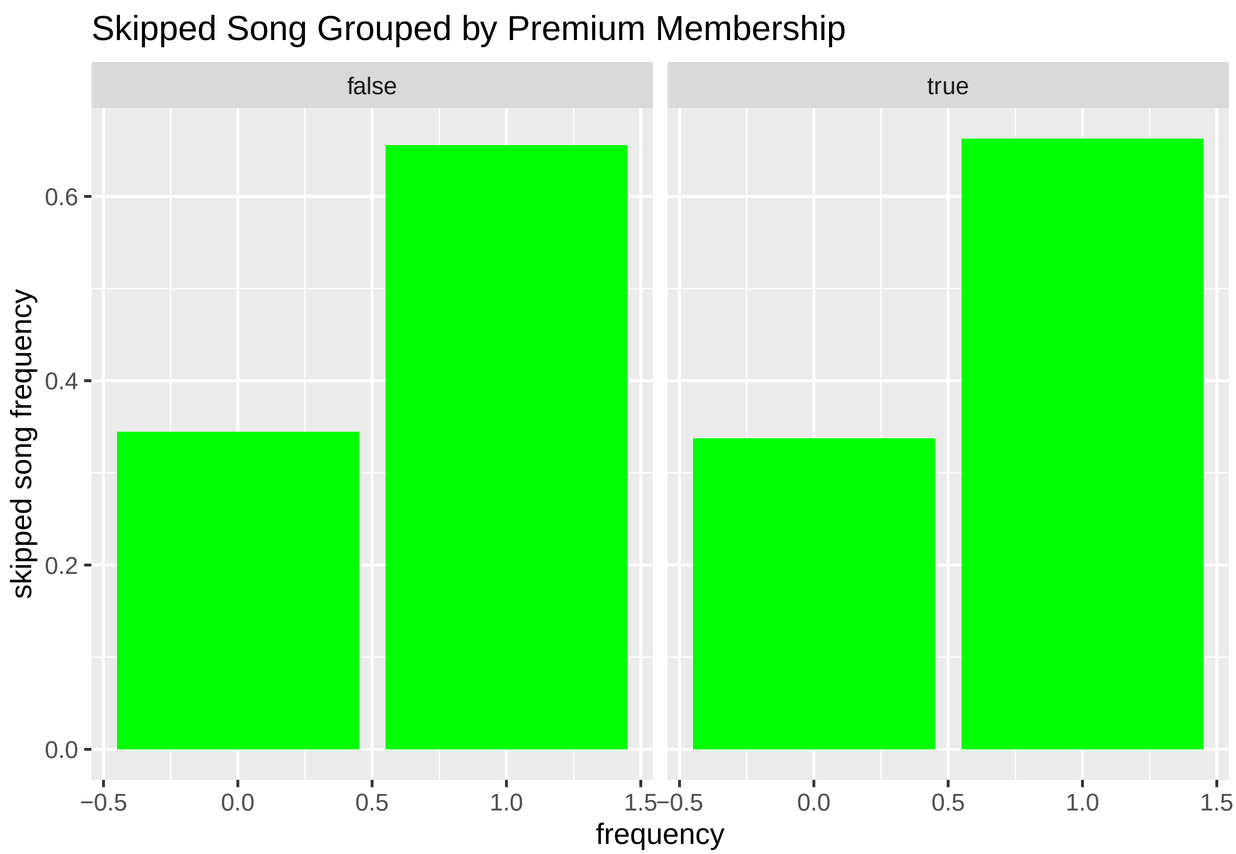
```
ggplot(df4) + geom_bar(aes(x = hour_of_day, y = ..prop.., group = premium),
                       fill = "green") +
```

```
  facet_wrap(~ premium ) +
  labs(title = "Hour of Day Frequency Grouped by Premium Membership") +
  xlab("frequency") + ylab("hour in military time")
```



We can see from this graph that users without premium are more likely to listen to music at night compared to premium listeners, and music with premium are more likely to listen to music in the afternoon than users without premium.

```
ggplot(df4) +
  geom_bar(aes(x = skipped, y = ..prop.., group = premium), fill = "green") +
  facet_wrap(~ premium ) +
  labs(title = "Skipped Song Grouped by Premium Membership") +
  xlab("frequency") + ylab("skipped song frequency")
```



We can see from this graph that users without premium skip songs the just as frequently as users with premium.

How likely is a person to skip a song?

```
sum(df4$skipped) / length(df4$skipped)
```

```
## [1] 0.66131
```

Users will skip a song about 66% of the time. This means that a model that guesses 'skipped' every time will have a 66% accuracy. We can use this as a baseline accuracy; any models less accurate than this are not helpful.

Machine Learning Algorithms

Separating Training and Testing Data

I first separated the data into training and testing data. The training data has 80000 rows (80% of the sample data) and the testing data has 20000 rows (20% of the sample data).

```
set.seed(1234)
i <- sample(1:nrow(df4), nrow(df4)*0.8, replace=FALSE)
train <- df4[i,]
test <- df4[-i,]
```

Next, I created three different versions of the training data. The first one, train_wo_date and test_wo_date, simply remove the date attribute from the original data. The next two split the remaining attributes into two different types - session attributes and user attributes. Session attributes, such as session position, context switch, and hour of day, are attributes describing the particular listening session that the instance comes from. The uesr attributes, such as historical behavior and premium status, are specific to the user who is listening.

```
train_wo_date <- cbind(train[,1:10], train[,12:16])
test_wo_date <- cbind(test[,1:10], test[,12:16])

train_session_data <- cbind(train[,1:6], train[,10], train[,13], train[,16])
names(train_session_data) <- c("session_position", "session_length",
                               "context_switch", "no_pause_before_play",
                               "short_pause_before_play",
                               "long_pause_before_play", "hour_of_day",
                               "context_type", "skipped")
test_session_data <- cbind(test[,1:6], test[,10], test[,13], test[,16])
names(test_session_data) <- c("session_position", "session_length",
                              "context_switch", "no_pause_before_play",
                              "short_pause_before_play",
                              "long_pause_before_play", "hour_of_day",
                              "context_type", "skipped")

train_user_data <- cbind(train[,7:9], train[,12], train[,14:16])
names(train_user_data) <- c("hist_user_behavior_n_seekfwd",
                            "hist_user_behavior_n_seekback",
                            "hist_user_behavior_is_shuffle", "premium",
                            "hist_user_behavior_reason_start",
                            "hist_user_behavior_reason_end", "skipped")
test_user_data <- cbind(test[,7:9], test[,12], test[,14:16])
names(test_user_data) <- c("hist_user_behavior_n_seekfwd",
                           "hist_user_behavior_n_seekback",
                           "hist_user_behavior_is_shuffle", "premium",
                           "hist_user_behavior_reason_start",
                           "hist_user_behavior_reason_end", "skipped")
```

Logistic Regression

I want to create a logistic regression model to predict whether a song was skipped. First, I want to try to predict this using attributes of the listening session.

```
glm1 <- glm(skipped~., data = train_session_data)
probability <- predict(glm1, newdata=test_session_data, type="response")
prediction = probability > 0.5
pred <- unname(prediction)
print(paste("Accuracy:    ", sum(pred == test_session_data$skipped) / 20000))
```

```
## [1] "Accuracy:     0.66395"
```

```
print(paste("Sensitivity: ", sum(pred == TRUE &
                                   test_session_data$skipped == TRUE) /
              sum(test_session_data$skipped == TRUE))) # sensitivity
```

```
## [1] "Sensitivity:  0.987462235649547"
```

```
print(paste("Specificity: ", sum(pred == FALSE &
                                   test_session_data$skipped == FALSE) /
              sum(test_session_data$skipped == FALSE))) # specificity
```

```
## [1] "Specificity:  0.0303254437869822"
```

This model has about a 66% accuracy, which is the same accuracy as a model that always guesses true. Therefore, we can conclude that the variables about the Spotify session are not good predictors. Although the model has a high sensitivity, it has a very low specificity, meaning there are a lot of false positives. Since the session variables are not good predictors, I will next try a model with the user attributes.

```
glm2 <- glm(skipped~., data = train_user_data)
probability <- predict(glm2, newdata=test_user_data, type="response")
prediction = probability > 0.5
pred <- unname(prediction)
print(paste("Accuracy:    ", sum(pred == test_session_data$skipped) / 20000))
```

```
## [1] "Accuracy:     0.98355"
```

```
print(paste("Sensitivity: ", sum(pred == TRUE &
                                   test_session_data$skipped == TRUE) /
              sum(test_session_data$skipped == TRUE)))
```

```
## [1] "Sensitivity:  0.978776435045317"
```

```
print(paste("Specificity: ", sum(pred == FALSE &
                                   test_session_data$skipped == FALSE) /
              sum(test_session_data$skipped == FALSE)))
```

```
## [1] "Specificity:  0.992899408284024"
```

This model, compared to the last one, performs a lot better. Although the sensitivity is about .01 lower, the specificity is .96 higher, and the accuracy is 98%. This indicates that the variables about the user are much better predictors than the variables about the session.

```
glm3 <- glm(skipped~., data = train_wo_date)
probability <- predict(glm3, newdata=test_wo_date, type="response")
prediction = probability > 0.5
pred <- unname(prediction)
print(paste("Accuracy:    ", sum(pred == test_session_data$skipped) / 20000))
```

```
## [1] "Accuracy:     0.98355"
```

```
print(paste("Sensitivity: ", sum(pred == TRUE &
                                   test_session_data$skipped == TRUE) /
              sum(test_session_data$skipped == TRUE)))
```

```
## [1] "Sensitivity:  0.978776435045317"
```

```
print(paste("Specificity: ", sum(pred == FALSE &
                                   test_session_data$skipped == FALSE) /
              sum(test_session_data$skipped == FALSE)))
```

```
## [1] "Specificity:  0.992899408284024"
```

The accuracy, sensitivity, and specificity are the exact same for the logistic regression model dependent on the user attributes and the logistic regression model dependent on all attributes. Adding the session attributes did not improve these metrics, once again supporting the idea that session attributes are bad predictors.

Naive Bayes

```
library(e1071)
library(caret)
```

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

```r
# data cleaning - Naive Bayes prefers integers in the factor format
train1 <- train_wo_date
test1 <- test_wo_date
train1$session_position <- factor(train_wo_date$session_position)
test1$session_position <- factor(test_wo_date$session_position)
train1$session_length <- factor(train_wo_date$session_length)
test1$session_length <- factor(test_wo_date$session_length)
train1$hist_user_behavior_n_seekfwd <-
  factor(train_wo_date$hist_user_behavior_n_seekfwd)
test1$hist_user_behavior_n_seekfwd <-
  factor(test_wo_date$hist_user_behavior_n_seekfwd)
train1$hist_user_behavior_n_seekback <-
  factor(train_wo_date$hist_user_behavior_n_seekback)
test1$hist_user_behavior_n_seekback <-
  factor(test_wo_date$hist_user_behavior_n_seekback)
test1$hour_of_day <- factor(test_wo_date$hour_of_day)
train1$hour_of_day <- factor(train_wo_date$hour_of_day)

nb1 <- naiveBayes(skipped~., data = train1, laplace = 1)
pred <- predict(nb1, newdata=test1, type = "class")
print(paste("Accuracy:    ", sum(pred == test_session_data$skipped) / 20000))
```

```
## [1] "Accuracy:     0.98415"
```

```r
print(paste("Sensitivity: ", sum(pred == 1 & test_session_data$skipped == 1) /
              sum(test_session_data$skipped == 1)))
```

```
## [1] "Sensitivity:  0.980135951661631"
```

```r
print(paste("Specificity: ", sum(pred == 0 & test_session_data$skipped == 0) /
              sum(test_session_data$skipped == 0)))
```

```
## [1] "Specificity:  0.992011834319527"
```

This Naive Bayes model works extremely well, with a 98.4% accuracy, a 0.980 sensitivity, and a 0.992 specificity.

Decision Tree

```r
library(tree)

# Data Cleaning
train2 <- train_wo_date
train2$skipped <- factor(train2$skipped)
test2 <- test_wo_date
test2$skipped <- factor(test2$skipped)

tree1 <- tree(skipped~., data = train2)
pred <- predict(tree1, newdata = test2, type = "class")
print(paste("Accuracy:    ", sum(pred == test_session_data$skipped) / 20000))
```

```
## [1] "Accuracy:     0.98845"
```

```r
print(paste("Sensitivity: ", sum(pred == 1 & test_session_data$skipped == 1) /
              sum(test_session_data$skipped == 1)))
```

```
## [1] "Sensitivity:  0.987990936555891"
```

```r
print(paste("Specificity: ", sum(pred == 0 & test_session_data$skipped == 0) /
              sum(test_session_data$skipped == 0)))
```

```
## [1] "Specificity:  0.989349112426036"
```

The Decision Tree model works very well, just like the other two, with a 98.8 accuracy, a 0.988 sensitivity, and a 0.989 specificity.

Algorithm Analysis

First, I would like the consider the first two logistic regression models. We discovered that the variables describing the user are much better predictors than the variables describing this session. This suggests whether or not a song is skipped is dependent more on the user's habits, not the situation surrounding the song in that moment. Some users inherently skip songs more than others; this insight means that to reduce the number of songs skipped, you must alter the user's habits.

Next, let us consider the three models that use all of the variables as predictors. The accuracy for these models ranges from 98.4 to 98.8, just a 0.4% difference. This means that all of the models have about the same accuracy. However, if we must rank them based on this minimal difference in accuracy, we can say that the decision tree is the most accurate, while naive bayes is second and logistic regression is last. Looking at the sensitivity and specificity, we can see that all 3 algorithms have a slightly higher sensitivity than specificity, meaning they are slightly more likely to have a false negative than a false negative than a false positive. Naive Bayes has the highest sensitivity, and logistic regression has the highest specificity. This does not mean that Naive Bayes or Logistic Regression is better; it highly depends on the circumstance. In some situations, we may prioritize sensitivity over specificity or vice versa.

So in conclusion, although all three models are almost equally good, we could rank them as follows:

1. Decision Tree
2. Naive Bayes
3. Logistic Regression