Task_1

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```
library(cluster)
library(VIM)
```

```
## Loading required package: colorspace
```

```
## Loading required package: grid
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
```

```
##     transpose
```

```
## VIM is ready to use.
##  Since version 4.0.0 the GUI is in its own package VIMGUI.
##
##           Please use the package to use the new (and old) GUI.
```

```
## Suggestions and bug-reports can be submitted at: https://github.com/alexkowa/VIM/issues
```

```
##
## Attaching package: 'VIM'
```

```
## The following object is masked from 'package:datasets':
##
##     sleep
```

```
library(fpc)
library(leaps)
library(ISLR)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```
## Loaded glmnet 3.0-1
```

```
library(ggvis)
```

```
##
## Attaching package: 'ggvis'
```

```
## The following object is masked from 'package:Matrix':
##
##     band
```

```
## The following object is masked from 'package:ggplot2':
##
##     resolution
```

```
set.seed(666)
```

```
spotify.clustering <- read_csv("inst/edited_spotify.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Artist = col_character(),
##   ArtistID = col_character(),
##   ArtistGenres = col_character(),
##   AlbumName = col_character(),
##   AlbumID = col_character(),
##   AlbumBestChartPosition = col_character(),
##   AlbumReleaseDate = col_character(),
##   TrackName = col_character(),
##   TrackID = col_character()
```

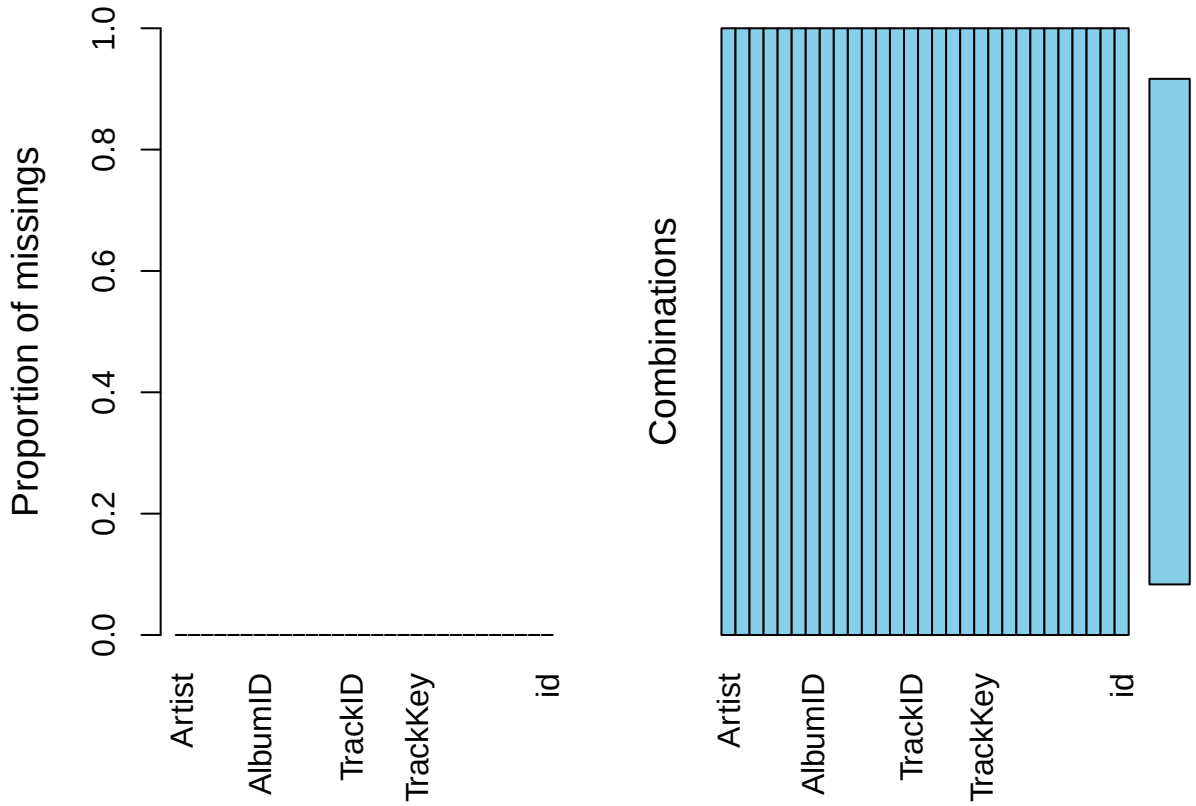
```
## )
```

```
## See spec(...) for full column specifications.
```

```
clean_data <- spotify.clustering %>%
  mutate(AlbumReleaseDate = parse_date_time(AlbumReleaseDate, orders = c("y", "ym","ymd"))) %>%
  #Old-school grepl method
  mutate(Artist = ifelse(grepl("Beyonc*", Artist), 'Beyonce', Artist)) %>%
  #Tidyverse str_detect method
  mutate(Artist = ifelse(Artist %>%
                           str_detect("Janelle Mon*"), 'Janelle Monae', Artist)) %>%
  mutate(AlbumBestChartPosition = ifelse(AlbumBestChartPosition %>%
                  #Assumption if chart position is #NA it never made it to the charts
                           str_detect("#N/A"), 0, AlbumBestChartPosition)) %>%
    mutate(AlbumBestChartPosition= as.numeric(AlbumBestChartPosition)) %>%
  na.omit() %>%
  mutate(id = row_number()) %>%
  mutate(id = as.character(id))
sapply(data, class)
```

```
##       ...      list   package   lib.loc   verbose     envir overwrite
##    "name"    "call"    "NULL"    "NULL"    "call"    "name" "logical"
##
##       "{"
```

```
aggr(clean_data) # checks for missing data
```



```
test_data <- subset(clean_data, ((AlbumName == "A Girl Called Dusty") |
                                 (AlbumName == "Action!") |
                                 (AlbumName == "Selling England By The Pound") |
                                 (AlbumName == "Carpenters") |
                                 (AlbumName == "Ride On") |
                                 (AlbumName == "Autoamerican") |
                                 (AlbumName == "Selected Ambient Works 85-92") |
                                 (AlbumName == "Different Class") |
                                 (AlbumName == "O") |
                                 (AlbumName == "The Elder Scrolls IV: Oblivion: Original Game Soundtrack
                                 (AlbumName == "AM") |
                                 (AlbumName == "An Awesome Wave")))

training_data <- subset(clean_data, ((AlbumName != "A Girl Called Dusty") &
                                     (AlbumName != "Action!") &
                                     (AlbumName != "Selling England By The Pound") &
                                     (AlbumName != "Carpenters") &
                                     (AlbumName != "Ride On") &
                                     (AlbumName != "Autoamerican") &
                                     (AlbumName != "Selected Ambient Works 85-92") &
                                     (AlbumName != "Different Class") &
                                     (AlbumName != "O") &
                                     (AlbumName != "The Elder Scrolls IV: Oblivion: Original Game Soundt
                                     (AlbumName != "AM") &
                                     (AlbumName != "An Awesome Wave")))

training_data_subsetted <- training_data[c("TrackDuration", "TrackDanceability",
                                 "TrackEnergy", "TrackKey", "TrackLoudness",
                                 "TrackSpeechiness", "TrackAcousticness",
                                 "TrackInstrumentalness", "TrackLiveness", "TrackValence",
                                 "TrackTempo")]
```

A new popularity variable needs to be defined to take into account all the other popularity variables

```
#Future maybe not implement due to time but could functionalise this and then use k fold cross validati
#Looking at implementing new variables for each popularity variable
#Can then take this and normalise to give a score out of 100.
#To start with the whole training dataset will be used
normy <- function(x){
  as.numeric(x)
  quantile(x, c(0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1), na.rm = TRUE)
}
#Assumption: NA means it didn't reach the charts, lower the position the better
abcp_normy <- normy(training_data$AlbumBestChartPosition)
anf_normy <- normy(training_data$ArtistNumFollowers)
awoc_normy <- normy(training_data$AlbumWeeksOnChart)
awn1_normy <- normy(training_data$AlbumWeeksNumberOne)
ap_normy <- normy(training_data$AlbumPopularity)
artp_normy <- normy(training_data$ArtistPopularity)

#Gets the 20% percentile for example
as.numeric(abcp_normy[6])
```

```
## [1] 3
```

```r
training_data <- training_data %>%
  mutate(album_bcp = if_else(AlbumBestChartPosition == 0, 1,
                         if_else(AlbumBestChartPosition == 1, 10,
                         if_else(AlbumBestChartPosition == 2, 9,
                         if_else(AlbumBestChartPosition == 3, 8,
                           0 ))))) %>% #... repeat for other variables
  mutate(artist_follow_n = 0) %>%
  mutate(album_pop_n = 0) %>%
  mutate(artist_pop_n = 0) %>%
  mutate(album_weeks_on_chart_n = if_else(AlbumWeeksOnChart == 0, 1, 0)) %>%
  mutate(album_weeks_number_1n = if_else(AlbumWeeksNumberOne == 0, 1, 0)) %>%
  mutate(popularity = -1) %>%
  mutate(AlbumReleaseDate = as.numeric(format(AlbumReleaseDate, "%y"))) %>%
  mutate(years_since_release = if_else(between(AlbumReleaseDate, 60, 99), (19 + (100 - AlbumReleaseDate
                                       (19 - AlbumReleaseDate)))


#album weeks number 1, this is fine as we are going > 0 and =< 1
for(i in 8:10){
  for(j in 1:nrow(training_data)){
      if(training_data$AlbumWeeksNumberOne[j] > as.numeric(awn1_normy[i]) &&
         training_data$AlbumWeeksNumberOne[j] <= as.numeric(awn1_normy[i + 1])){
    training_data$album_weeks_number_1n[j] = i
  }
  }
}

for(i in 2:10){
  for(j in 1:nrow(training_data)){
      if(training_data$AlbumWeeksOnChart[j] > as.numeric(awoc_normy[i]) &&
         training_data$AlbumWeeksOnChart[j] <= as.numeric(awoc_normy[i + 1])){
    training_data$album_weeks_on_chart_n[j] = i
  }
  }
}

#album weeks on chart

#albumchartposition
for(i in 6:10){
  for(j in 1:nrow(training_data)){
      if(training_data$AlbumBestChartPosition[j] > as.numeric(abcp_normy[i]) &&
         training_data$AlbumBestChartPosition[j] <= as.numeric(abcp_normy[i + 1])){
    training_data$album_bcp[j] = (12-i)
  }
  }
}

#Mutate at end to sort out end case
for(i in 1:10){
  for(j in 1:nrow(training_data)){
      if(training_data$ArtistNumFollowers[j] >= as.numeric(anf_normy[i]) &&
         training_data$ArtistNumFollowers[j] < as.numeric(anf_normy[i + 1])){
```

```
    training_data$artist_follow_n[j] = i
  }
  }
}
#Create and update artist_popularity
for(i in 1:10){
  for(j in 1:nrow(training_data)){
      if(training_data$ArtistPopularity[j] >= as.numeric(artp_normy[i]) &&
         training_data$ArtistPopularity[j] < as.numeric(artp_normy[i + 1])){
    training_data$artist_pop_n[j] = i
  }
  }
}
#Create and update album popularity
for(i in 1:10){
  for(j in 1:nrow(training_data)){
      if(training_data$AlbumPopularity[j] >= as.numeric(ap_normy[i]) &&
         training_data$AlbumPopularity[j] < as.numeric(ap_normy[i + 1])){
    training_data$album_pop_n[j] = i
  }
  }
}
```

Defining a function to summarise the data input and define the y variable

```
spotify_summarise <- function(x){
  x %>%
     group_by(Artist, AlbumName, AlbumReleaseDate) %>%
  summarise(track_duration_mean = mean(TrackDuration),
            track_duration_IQR = IQR(TrackDuration),
            track_danceability_mean = mean(TrackDanceability),
            track_danceability_IQR = IQR(TrackDanceability),
            track_energy_mean = mean(TrackEnergy),
            track_energy_IQR = IQR(TrackEnergy),
            track_loudness_mean = mean(TrackLoudness),
            track_loudness_IQR = IQR(TrackLoudness),
            track_speechiness_mean = mean(TrackSpeechiness),
            track_speechiness_IQR = IQR(TrackSpeechiness),
            track_acousticness_mean = mean(TrackAcousticness),
            track_acousticness_IQR = IQR(TrackAcousticness),
            track_instrumentalness_mean = mean(TrackInstrumentalness),
            track_instrumentalness_IQR = IQR(TrackInstrumentalness),
            track_valence_mean = mean(TrackValence),
            track_valence_IQR = IQR(TrackValence),
            track_tempo_mean = mean(TrackTempo),
            track_tempo_IQR = IQR(TrackTempo),
            popularity = mean(popularity)

  )
}

define_y <- function(y){
y %>%
  select(popularity) %>%
```

```
  unlist() %>%
  as.numeric() %>%
  na.omit()
}
```

First Popularity Combination

```
#Could maybe add a years from release as a penalty to make newer songs higher rated or subset the data?
#Should probably add some sort of verification to the numbers for now they are arbitrary?
#Tried 2 and 5 for the multiplcation parameter in the years since release scaling
#But these both resulted in higher MSE
training_data_pop_1 <- training_data %>%
  mutate(popularity = ((25 * training_data$album_pop_n + 25 * training_data$artist_pop_n +
                          25 * training_data$artist_follow_n + 10 * training_data$album_bcp +
                          10 * training_data$album_weeks_on_chart_n +
                          5 * training_data$album_weeks_number_1n)/(
                          10 * ((100 + 3 * training_data$years_since_release)/
                          (100 + training_data$years_since_release))))) %>%
  #Correcting for the max term which is not included in the if statement
  mutate(artist_follow_n = if_else(artist_follow_n == 0, 10, artist_follow_n)) %>%
  mutate(artist_pop_n = if_else(artist_pop_n == 0, 10, artist_pop_n)) %>%
  mutate(album_pop_n = if_else(album_pop_n == 0, 10, album_pop_n))


data_1 <- training_data_pop_1 %>%
  spotify_summarise %>%
  ungroup() %>%
  select(-AlbumName, -Artist, -AlbumReleaseDate)
```

Second Popularity Combination

```
#Could maybe add a years from release as a penalty to make newer songs higher rated or subset the data?
#Should probably add some sort of verification to the numbers for now they are arbitrary?
value <- 100/3
training_data_pop_2 <- training_data %>%
  mutate(popularity = ((value * training_data$album_pop_n + value * training_data$artist_pop_n +
                          value * training_data$artist_follow_n)/(
                          10 * ((100 + 3 * training_data$years_since_release)/
                          (100 + training_data$years_since_release))))) %>%
  #Correcting for the max term which is not included in the if statement
  mutate(artist_follow_n = if_else(artist_follow_n == 0, 10, artist_follow_n)) %>%
  mutate(artist_pop_n = if_else(artist_pop_n == 0, 10, artist_pop_n)) %>%
  mutate(album_pop_n = if_else(album_pop_n == 0, 10, album_pop_n))


data_2 <- training_data_pop_2 %>%
  spotify_summarise %>%
  ungroup() %>%
  select(-AlbumName, -Artist, -AlbumReleaseDate)
```

Third Popularity Combination

```
#Could maybe add a years from release as a penalty to make newer songs higher rated or subset the data?
#Should probably add some sort of verification to the numbers for now they are arbitrary?
value <- 100/6
training_data_pop_3 <- training_data %>%
```

```
  mutate(popularity = ((value * training_data$album_pop_n + value * training_data$artist_pop_n +
                          value * training_data$artist_follow_n + value * training_data$album_bcp +
                          value * training_data$album_weeks_on_chart_n +
                          value * training_data$album_weeks_number_1n)/(
                          10 * ((100 + 3 * training_data$years_since_release)/
                          (100 + training_data$years_since_release))))) %>%
  #Correcting for the max term which is not included in the if statement
  mutate(artist_follow_n = if_else(artist_follow_n == 0, 10, artist_follow_n)) %>%
  mutate(artist_pop_n = if_else(artist_pop_n == 0, 10, artist_pop_n)) %>%
  mutate(album_pop_n = if_else(album_pop_n == 0, 10, album_pop_n))


data_3 <- training_data_pop_3 %>%
  spotify_summarise %>%
  ungroup() %>%
  select(-AlbumName, -Artist, -AlbumReleaseDate)
```

Looking at the the three datasets using best subset regression: Dataset 1:

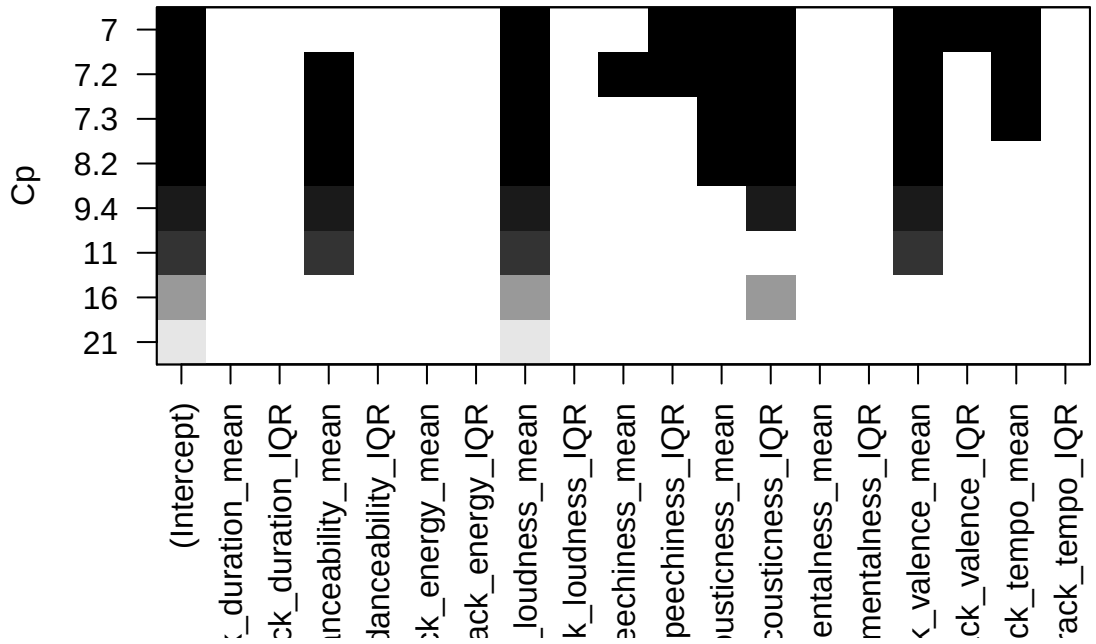
```
#all subsets regression


m <- regsubsets(popularity ~ ., data = data_1)

summary(m)
```

```
## Subset selection object
## Call: regsubsets.formula(popularity ~ ., data = data_1)
## 18 Variables  (and intercept)
##                              Forced in Forced out
## track_duration_mean              FALSE      FALSE
## track_duration_IQR               FALSE      FALSE
## track_danceability_mean          FALSE      FALSE
## track_danceability_IQR           FALSE      FALSE
## track_energy_mean                FALSE      FALSE
## track_energy_IQR                 FALSE      FALSE
## track_loudness_mean              FALSE      FALSE
## track_loudness_IQR               FALSE      FALSE
## track_speechiness_mean           FALSE      FALSE
## track_speechiness_IQR            FALSE      FALSE
## track_acousticness_mean          FALSE      FALSE
## track_acousticness_IQR           FALSE      FALSE
## track_instrumentalness_mean      FALSE      FALSE
## track_instrumentalness_IQR       FALSE      FALSE
## track_valence_mean               FALSE      FALSE
## track_valence_IQR                FALSE      FALSE
## track_tempo_mean                 FALSE      FALSE
## track_tempo_IQR                  FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          track_duration_mean track_duration_IQR track_danceability_mean
## 1  ( 1 ) " "                 " "                " "
## 2  ( 1 ) " "                 " "                " "
## 3  ( 1 ) " "                 " "                "*"
```

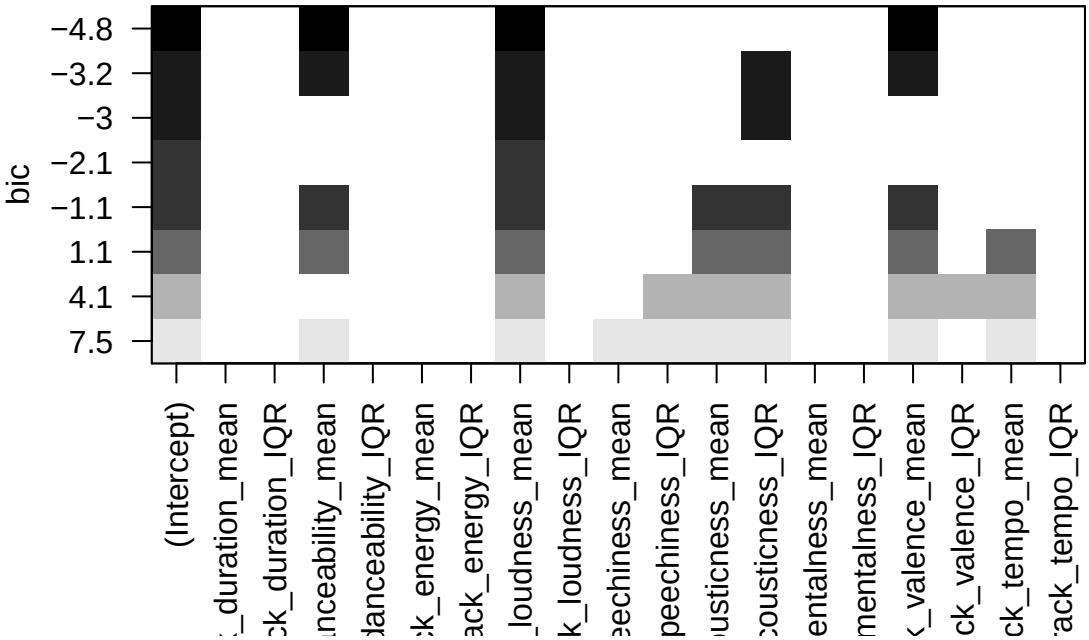
```
## 4  ( 1 ) " "                 " "                 "*"
## 5  ( 1 ) " "                 " "                 "*"
## 6  ( 1 ) " "                 " "                 "*"
## 7  ( 1 ) " "                 " "                 " "
## 8  ( 1 ) " "                 " "                 "*"
##          track_danceability_IQR track_energy_mean track_energy_IQR
## 1  ( 1 ) " "                   " "                 " "
## 2  ( 1 ) " "                   " "                 " "
## 3  ( 1 ) " "                   " "                 " "
## 4  ( 1 ) " "                   " "                 " "
## 5  ( 1 ) " "                   " "                 " "
## 6  ( 1 ) " "                   " "                 " "
## 7  ( 1 ) " "                   " "                 " "
## 8  ( 1 ) " "                   " "                 " "
##          track_loudness_mean track_loudness_IQR track_speechiness_mean
## 1  ( 1 ) "*"                 " "                 " "
## 2  ( 1 ) "*"                 " "                 " "
## 3  ( 1 ) "*"                 " "                 " "
## 4  ( 1 ) "*"                 " "                 " "
## 5  ( 1 ) "*"                 " "                 " "
## 6  ( 1 ) "*"                 " "                 " "
## 7  ( 1 ) "*"                 " "                 " "
## 8  ( 1 ) "*"                 " "                 "*"
##          track_speechiness_IQR track_acousticness_mean
## 1  ( 1 ) " "                   " "
## 2  ( 1 ) " "                   " "
## 3  ( 1 ) " "                   " "
## 4  ( 1 ) " "                   " "
## 5  ( 1 ) " "                   "*"
## 6  ( 1 ) " "                   "*"
## 7  ( 1 ) "*"                   "*"
## 8  ( 1 ) "*"                   "*"
##          track_acousticness_IQR track_instrumentalness_mean
## 1  ( 1 ) " "                    " "
## 2  ( 1 ) "*"                    " "
## 3  ( 1 ) " "                    " "
## 4  ( 1 ) "*"                    " "
## 5  ( 1 ) "*"                    " "
## 6  ( 1 ) "*"                    " "
## 7  ( 1 ) "*"                    " "
## 8  ( 1 ) "*"                    " "
##          track_instrumentalness_IQR track_valence_mean track_valence_IQR
## 1  ( 1 ) " "                        " "                 " "
## 2  ( 1 ) " "                        " "                 " "
## 3  ( 1 ) " "                        "*"                 " "
## 4  ( 1 ) " "                        "*"                 " "
## 5  ( 1 ) " "                        "*"                 " "
## 6  ( 1 ) " "                        "*"                 " "
## 7  ( 1 ) " "                        "*"                 "*"
## 8  ( 1 ) " "                        "*"                 " "
##          track_tempo_mean track_tempo_IQR
## 1  ( 1 ) " "              " "
## 2  ( 1 ) " "              " "
## 3  ( 1 ) " "              " "
```

```
## 4  ( 1 ) " "               " " "
## 5  ( 1 ) " "               " " "
## 6  ( 1 ) "*"               " " "
## 7  ( 1 ) "*"               " " "
## 8  ( 1 ) "*"               " " "
```

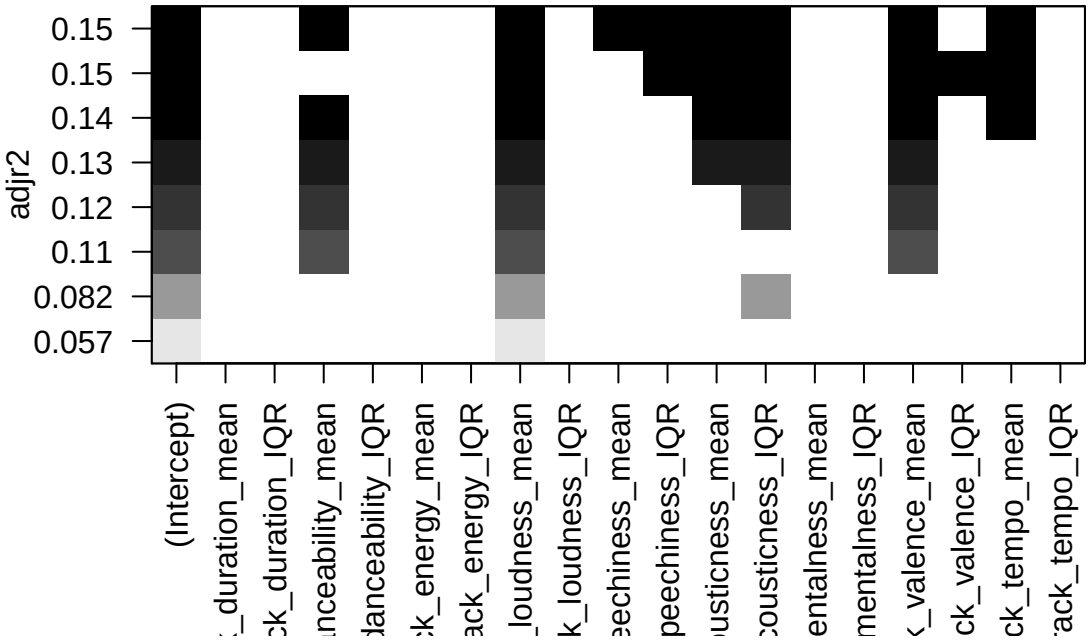
```
#measures
par(mfrow = c(1,1))

plot(m, scale = "Cp")
```




```
plot(m, scale = "bic")
```




```
plot(m, scale = "adjr2")
```




```
#check Adjusted R2
rsq <- as.data.frame(summary(m)$adjr2)
names(rsq) <- "Adjusted-R2"
rsq %>%
        ggvis(x=~ c(1:nrow(rsq)), y=~`Adjusted-R2` ) %>%
        layer_points(fill = ~`Adjusted-R2`) %>%
        add_axis("y", title = "Adjusted-R2`") %>%
        add_axis("x", title = "Number of variables")
```

Renderer: SVG | Canvas

Download

```
#the final Adjusted R2 is very high

reg.summary <- summary(m)


#compare RSS, Cp, bic, adjr2
par(mfrow=c(2,2))
plot(reg.summary$rss ,xlab="Number of Variables ",ylab="RSS",type="l")
plot(reg.summary$adjr2 ,xlab="Number of Variables ", ylab="Adjusted RSq",type="l")
which.max(reg.summary$adjr2)
```

```
## [1] 8
```

```
points(which.max(reg.summary$adjr2),reg.summary$adjr2[which.max(reg.summary$adjr2)], col="red",cex=2,pch
plot(reg.summary$cp ,xlab="Number of Variables ",ylab="Cp", type='l')
```

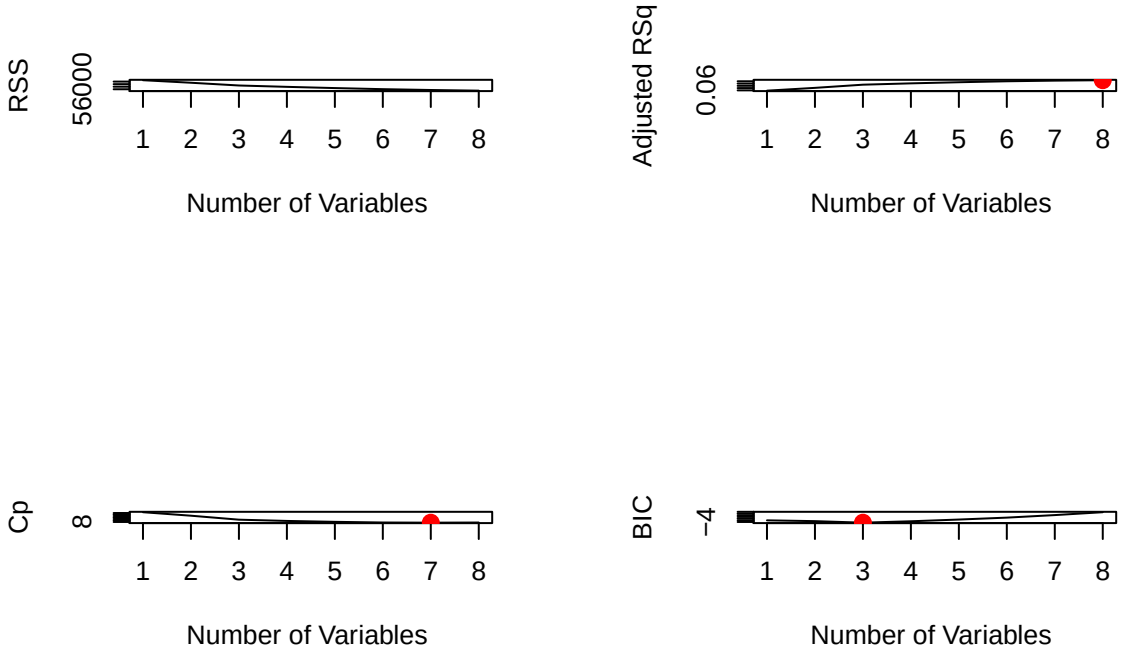
```
which.min(reg.summary$cp )
```

```
## [1] 7
```

```
points(which.min(reg.summary$cp ),reg.summary$cp [which.min(reg.summary$cp )],col="red",cex=2,pch=20)
plot(reg.summary$bic ,xlab="Number of Variables ",ylab="BIC",type='l')
which.min(reg.summary$bic )
```

```
## [1] 3
```

```
points(which.min(reg.summary$bic ),reg.summary$bic [which.min(reg.summary$bic )],col="red",cex=2,pch=20)
```




```
par(mfrow=c(1,1))


coef(m, 8)
```

```
##             (Intercept) track_danceability_mean     track_loudness_mean
##               83.9910105               19.9789419                1.4139635
##  track_speechiness_mean   track_speechiness_IQR track_acousticness_mean
##              -63.7749607               88.2040329              -13.0000750
##  track_acousticness_IQR      track_valence_mean        track_tempo_mean
##               15.6446656              -23.7369274               -0.2280043
```

```
#a selected best model by adjr2
m1 <- lm( popularity ~  track_loudness_mean  + track_speechiness_mean+ track_speechiness_IQR+ track_acou
summary(m1)
```

```
##
```

```
## Call:
## lm(formula = popularity ~ track_loudness_mean + track_speechiness_mean +
##     track_speechiness_IQR + track_acousticness_mean + track_acousticness_IQR +
##     track_valence_mean + track_valence_IQR + track_tempo_mean,
##     data = data_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.461 -12.614  -1.311  12.561  41.417
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               89.7982    17.6714   5.082 8.99e-07 ***
## track_loudness_mean        1.4621     0.4467   3.273  0.00126 **
## track_speechiness_mean   -49.8739    41.2521  -1.209  0.22818
## track_speechiness_IQR     85.9163    44.6818   1.923  0.05601 .
## track_acousticness_mean  -13.7498     6.6996  -2.052  0.04152 *
## track_acousticness_IQR    15.8445     7.4956   2.114  0.03585 *
## track_valence_mean       -17.8095     7.8587  -2.266  0.02458 *
## track_valence_IQR         15.6767    11.1007   1.412  0.15953
## track_tempo_mean          -0.2513     0.1241  -2.025  0.04427 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.1 on 188 degrees of freedom
## Multiple R-squared:  0.1824, Adjusted R-squared:  0.1476
## F-statistic: 5.243 on 8 and 188 DF,  p-value: 6.211e-06
```

```
#predictions on testing data
preds <- predict(m1, newdata = data_1 )
mse_bs1 <- mean((preds - data_1$popularity)^2)
absolute_bs1 <- sqrt( mean((preds - data_1$popularity)^2) )
```

Dataset 2:

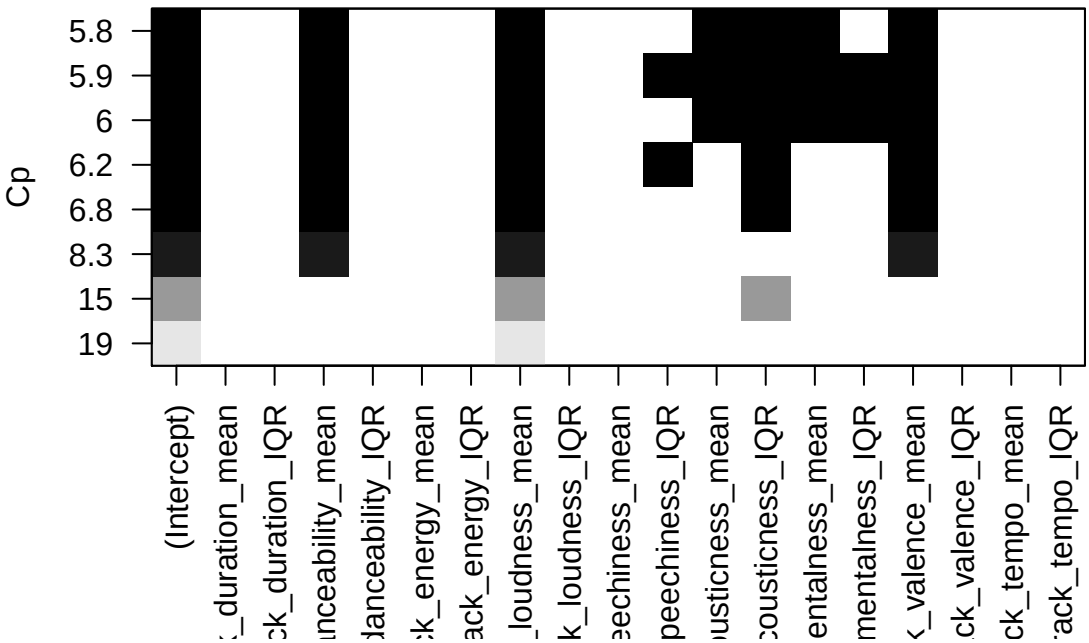
```
#all subsets regression


m <- regsubsets(popularity ~ ., data = data_2)

summary(m)
```

```
## Subset selection object
## Call: regsubsets.formula(popularity ~ ., data = data_2)
## 18 Variables  (and intercept)
##                             Forced in Forced out
## track_duration_mean             FALSE      FALSE
## track_duration_IQR              FALSE      FALSE
## track_danceability_mean         FALSE      FALSE
## track_danceability_IQR          FALSE      FALSE
## track_energy_mean               FALSE      FALSE
## track_energy_IQR                FALSE      FALSE
## track_loudness_mean             FALSE      FALSE
## track_loudness_IQR              FALSE      FALSE
## track_speechiness_mean          FALSE      FALSE
## track_speechiness_IQR           FALSE      FALSE
```

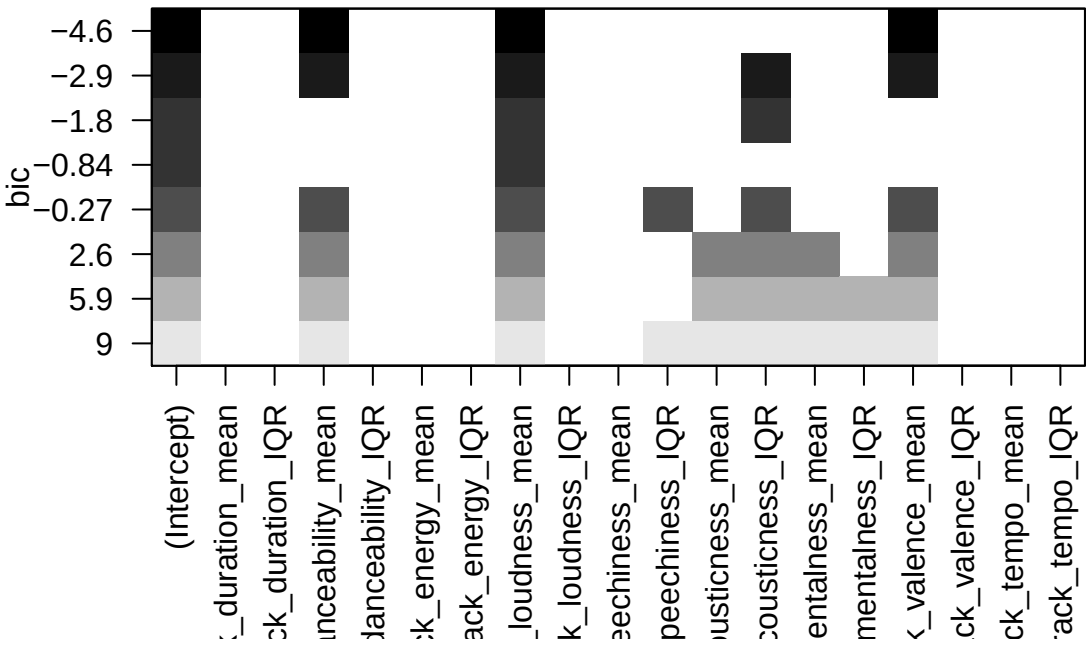
```
## track_acousticness_mean         FALSE      FALSE
## track_acousticness_IQR          FALSE      FALSE
## track_instrumentalness_mean     FALSE      FALSE
## track_instrumentalness_IQR      FALSE      FALSE
## track_valence_mean              FALSE      FALSE
## track_valence_IQR               FALSE      FALSE
## track_tempo_mean                FALSE      FALSE
## track_tempo_IQR                 FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          track_duration_mean track_duration_IQR track_danceability_mean
## 1  ( 1 ) " "                 " "                " "
## 2  ( 1 ) " "                 " "                " "
## 3  ( 1 ) " "                 " "                "*"
## 4  ( 1 ) " "                 " "                "*"
## 5  ( 1 ) " "                 " "                "*"
## 6  ( 1 ) " "                 " "                "*"
## 7  ( 1 ) " "                 " "                "*"
## 8  ( 1 ) " "                 " "                "*"
##          track_danceability_IQR track_energy_mean track_energy_IQR
## 1  ( 1 ) " "                    " "               " "
## 2  ( 1 ) " "                    " "               " "
## 3  ( 1 ) " "                    " "               " "
## 4  ( 1 ) " "                    " "               " "
## 5  ( 1 ) " "                    " "               " "
## 6  ( 1 ) " "                    " "               " "
## 7  ( 1 ) " "                    " "               " "
## 8  ( 1 ) " "                    " "               " "
##          track_loudness_mean track_loudness_IQR track_speechiness_mean
## 1  ( 1 ) "*"                 " "                " "
## 2  ( 1 ) "*"                 " "                " "
## 3  ( 1 ) "*"                 " "                " "
## 4  ( 1 ) "*"                 " "                " "
## 5  ( 1 ) "*"                 " "                " "
## 6  ( 1 ) "*"                 " "                " "
## 7  ( 1 ) "*"                 " "                " "
## 8  ( 1 ) "*"                 " "                " "
##          track_speechiness_IQR track_acousticness_mean
## 1  ( 1 ) " "                   " "
## 2  ( 1 ) " "                   " "
## 3  ( 1 ) " "                   " "
## 4  ( 1 ) " "                   " "
## 5  ( 1 ) "*"                   " "
## 6  ( 1 ) " "                   "*"
## 7  ( 1 ) " "                   "*"
## 8  ( 1 ) "*"                   "*"
##          track_acousticness_IQR track_instrumentalness_mean
## 1  ( 1 ) " "                    " "
## 2  ( 1 ) "*"                    " "
## 3  ( 1 ) " "                    " "
## 4  ( 1 ) "*"                    " "
## 5  ( 1 ) "*"                    " "
## 6  ( 1 ) "*"                    "*"
## 7  ( 1 ) "*"                    "*"
```

```
## 8  ( 1 ) "*"                       "*"
##           track_instrumentalness_IQR track_valence_mean track_valence_IQR
## 1  ( 1 ) " "                         " "                " "
## 2  ( 1 ) " "                         " "                " "
## 3  ( 1 ) " "                         "*"                " "
## 4  ( 1 ) " "                         "*"                " "
## 5  ( 1 ) " "                         "*"                " "
## 6  ( 1 ) " "                         "*"                " "
## 7  ( 1 ) "*"                         "*"                " "
## 8  ( 1 ) "*"                         "*"                " "
##           track_tempo_mean track_tempo_IQR
## 1  ( 1 ) " "              " "
## 2  ( 1 ) " "              " "
## 3  ( 1 ) " "              " "
## 4  ( 1 ) " "              " "
## 5  ( 1 ) " "              " "
## 6  ( 1 ) " "              " "
## 7  ( 1 ) " "              " "
## 8  ( 1 ) " "              " "
```

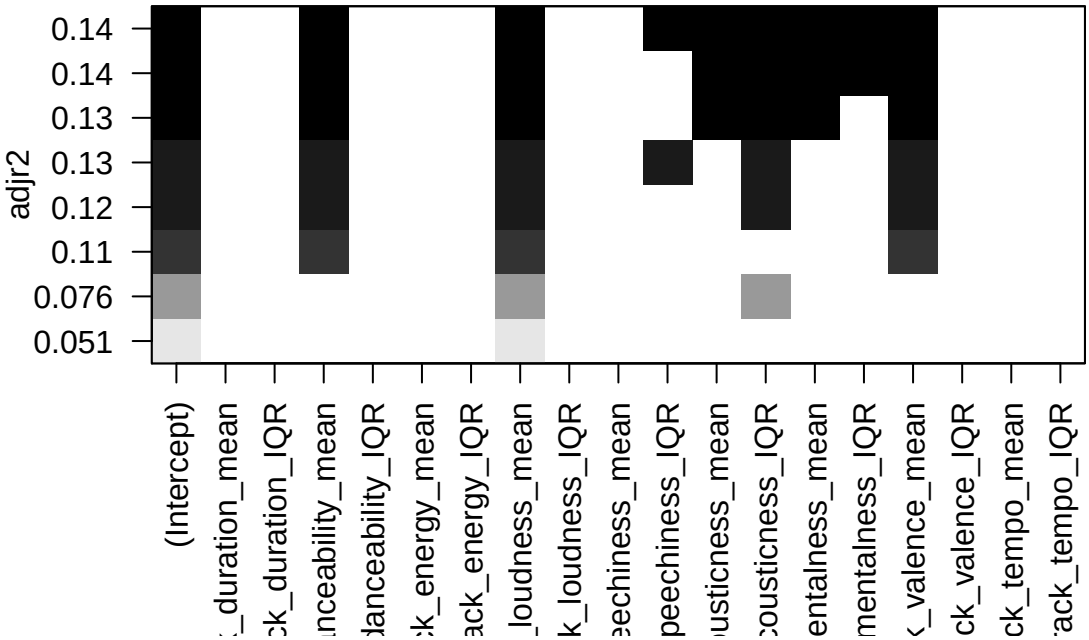
```r
#measures
par(mfrow = c(1,1))

plot(m, scale = "Cp")
```



```
plot(m, scale = "bic")
```




```
plot(m, scale = "adjr2")
```




```
#check Adjusted R2
rsq <- as.data.frame(summary(m)$adjr2)
names(rsq) <- "Adjusted-R2"
rsq %>%
        ggvis(x=~ c(1:nrow(rsq)), y=~`Adjusted-R2` ) %>%
        layer_points(fill = ~`Adjusted-R2`) %>%
        add_axis("y", title = "Adjusted-R2`") %>%
        add_axis("x", title = "Number of variables")
```

Renderer: SVG | Canvas

Download

```
#the final Adjusted R2 is very high

reg.summary <- summary(m)


#compare RSS, Cp, bic, adjr2
par(mfrow=c(2,2))
plot(reg.summary$rss ,xlab="Number of Variables ",ylab="RSS",type="l")
plot(reg.summary$adjr2 ,xlab="Number of Variables ", ylab="Adjusted RSq",type="l")
which.max(reg.summary$adjr2)
```

```
## [1] 8
```

```
points(which.max(reg.summary$adjr2),reg.summary$adjr2[which.max(reg.summary$adjr2)], col="red",cex=2,pch
plot(reg.summary$cp ,xlab="Number of Variables ",ylab="Cp", type='l')
```

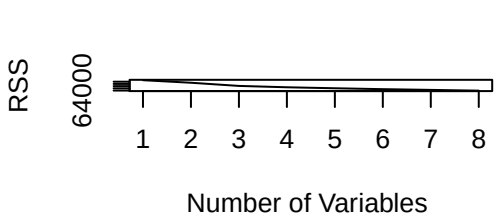
```
which.min(reg.summary$cp )
```

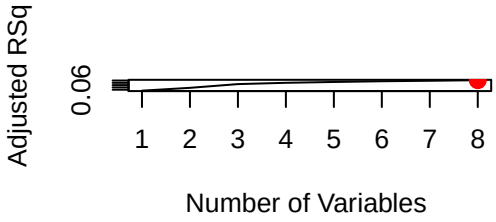
```
## [1] 6
```

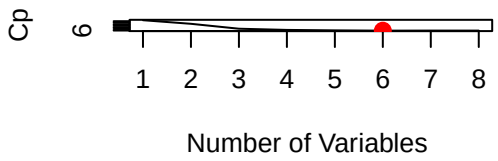
```
points(which.min(reg.summary$cp ),reg.summary$cp [which.min(reg.summary$cp )],col="red",cex=2,pch=20)
plot(reg.summary$bic ,xlab="Number of Variables ",ylab="BIC",type='l')
which.min(reg.summary$bic )
```

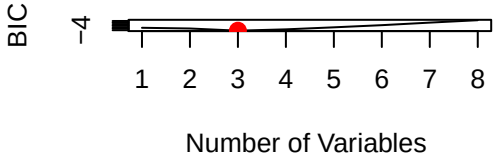
```
## [1] 3
```

```
points(which.min(reg.summary$bic ),reg.summary$bic [which.min(reg.summary$bic )],col="red",cex=2,pch=20)
```










```
par(mfrow=c(1,1))


coef(m, 8)
```

```
##                 (Intercept)     track_danceability_mean
##                   48.969495                   30.218862
##         track_loudness_mean       track_speechiness_IQR
##                    1.105621                   34.576292
##     track_acousticness_mean      track_acousticness_IQR
##                  -11.485357                   19.033778
## track_instrumentalness_mean  track_instrumentalness_IQR
##                  -18.275231                   14.165837
##          track_valence_mean
##                  -30.535470
```

```r
#a selected best model by adjr2
m1 <- lm( popularity ~  track_duration_IQR + track_danceability_mean + track_loudness_mean + track_spee
summary(m1)
```

```
##
## Call:
## lm(formula = popularity ~ track_duration_IQR + track_danceability_mean +
##     track_loudness_mean + track_speechiness_mean + track_speechiness_IQR +
##     track_acousticness_IQR + track_valence_mean + track_tempo_mean,
##     data = data_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.919 -15.244   0.904  14.473  45.271
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               7.488e+01  1.969e+01   3.803 0.000193 ***
## track_duration_IQR        2.032e-05  2.163e-05   0.939 0.348773
## track_danceability_mean   3.062e+01  1.410e+01   2.171 0.031160 *
## track_loudness_mean       1.984e+00  4.229e-01   4.692 5.19e-06 ***
## track_speechiness_mean   -7.212e+01  4.506e+01  -1.601 0.111163
## track_speechiness_IQR     1.008e+02  4.825e+01   2.090 0.037955 *
## track_acousticness_IQR    1.344e+01  8.000e+00   1.680 0.094703 .
## track_valence_mean       -2.279e+01  1.026e+01  -2.222 0.027494 *
## track_tempo_mean         -1.917e-01  1.338e-01  -1.434 0.153371
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.54 on 188 degrees of freedom
## Multiple R-squared:  0.168,  Adjusted R-squared:  0.1326
## F-statistic: 4.746 on 8 and 188 DF,  p-value: 2.539e-05
```

```r
#predictions on testing data
preds <- predict(m1, newdata = data_2)

absolute_bs2 <- sqrt( mean((preds - data_2$popularity)^2) )
mse_bs2 <- mean((preds - data_2$popularity)^2)
```

Third Dataset:

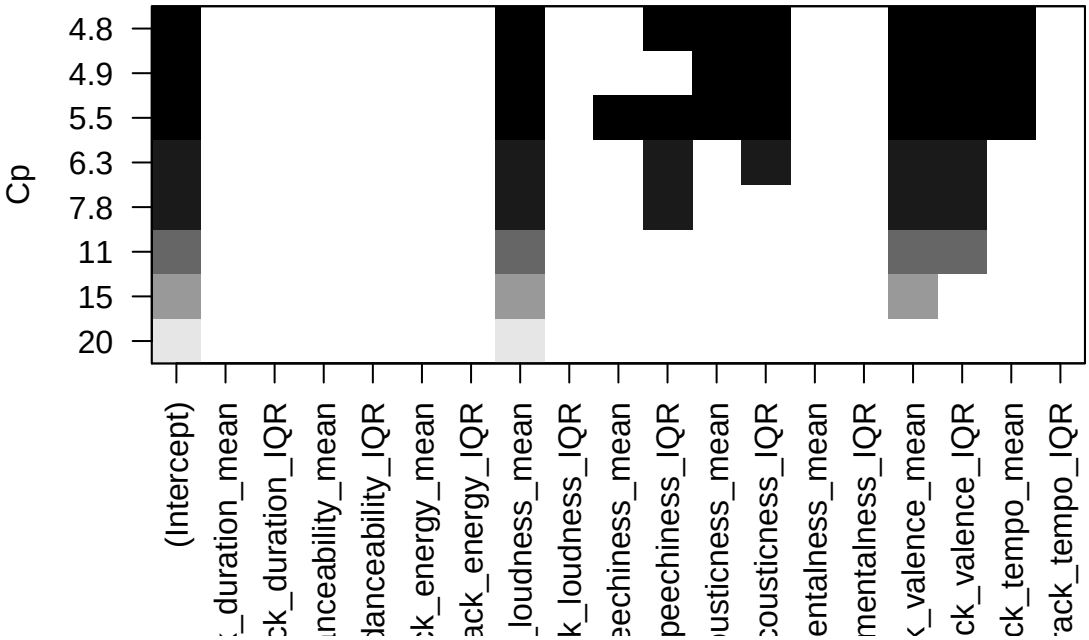
```r
m <- regsubsets(popularity ~ ., data = data_3)

summary(m)
```

```
## Subset selection object
## Call: regsubsets.formula(popularity ~ ., data = data_3)
## 18 Variables  (and intercept)
##                             Forced in Forced out
## track_duration_mean             FALSE      FALSE
## track_duration_IQR              FALSE      FALSE
## track_danceability_mean         FALSE      FALSE
## track_danceability_IQR          FALSE      FALSE
## track_energy_mean               FALSE      FALSE
## track_energy_IQR                FALSE      FALSE
```

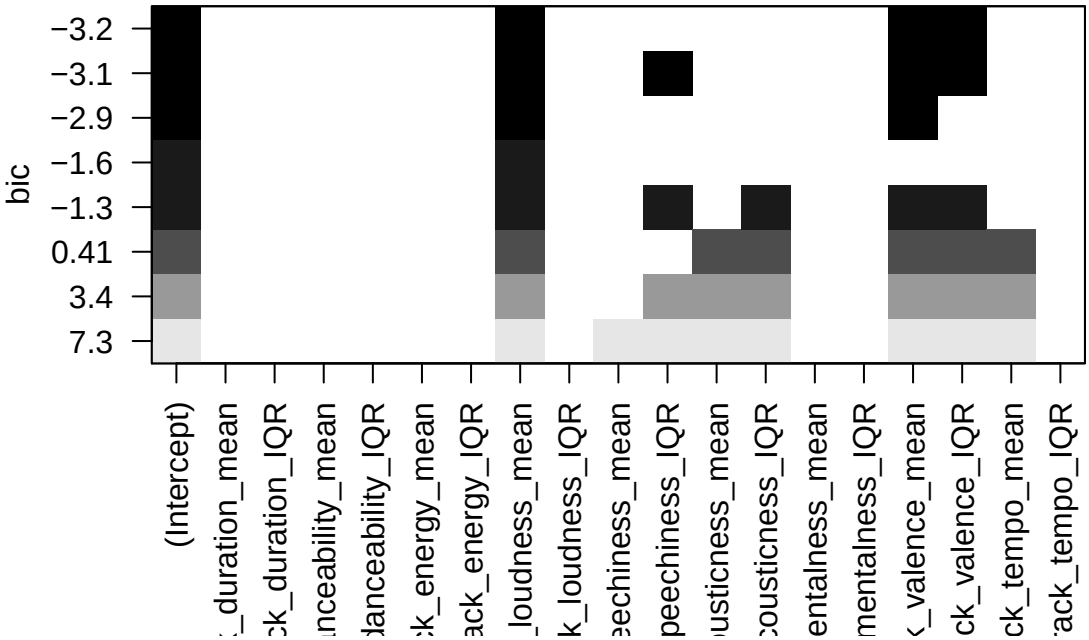
```
## track_loudness_mean             FALSE      FALSE
## track_loudness_IQR              FALSE      FALSE
## track_speechiness_mean          FALSE      FALSE
## track_speechiness_IQR           FALSE      FALSE
## track_acousticness_mean         FALSE      FALSE
## track_acousticness_IQR          FALSE      FALSE
## track_instrumentalness_mean     FALSE      FALSE
## track_instrumentalness_IQR      FALSE      FALSE
## track_valence_mean              FALSE      FALSE
## track_valence_IQR               FALSE      FALSE
## track_tempo_mean                FALSE      FALSE
## track_tempo_IQR                 FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          track_duration_mean track_duration_IQR track_danceability_mean
## 1  ( 1 ) " "                 " "                " "
## 2  ( 1 ) " "                 " "                " "
## 3  ( 1 ) " "                 " "                " "
## 4  ( 1 ) " "                 " "                " "
## 5  ( 1 ) " "                 " "                " "
## 6  ( 1 ) " "                 " "                " "
## 7  ( 1 ) " "                 " "                " "
## 8  ( 1 ) " "                 " "                " "
##          track_danceability_IQR track_energy_mean track_energy_IQR
## 1  ( 1 ) " "                    " "               " "
## 2  ( 1 ) " "                    " "               " "
## 3  ( 1 ) " "                    " "               " "
## 4  ( 1 ) " "                    " "               " "
## 5  ( 1 ) " "                    " "               " "
## 6  ( 1 ) " "                    " "               " "
## 7  ( 1 ) " "                    " "               " "
## 8  ( 1 ) " "                    " "               " "
##          track_loudness_mean track_loudness_IQR track_speechiness_mean
## 1  ( 1 ) "*"                 " "                " "
## 2  ( 1 ) "*"                 " "                " "
## 3  ( 1 ) "*"                 " "                " "
## 4  ( 1 ) "*"                 " "                " "
## 5  ( 1 ) "*"                 " "                " "
## 6  ( 1 ) "*"                 " "                " "
## 7  ( 1 ) "*"                 " "                " "
## 8  ( 1 ) "*"                 " "                "*"
##          track_speechiness_IQR track_acousticness_mean
## 1  ( 1 ) " "                   " "
## 2  ( 1 ) " "                   " "
## 3  ( 1 ) " "                   " "
## 4  ( 1 ) "*"                   " "
## 5  ( 1 ) "*"                   " "
## 6  ( 1 ) " "                   "*"
## 7  ( 1 ) "*"                   "*"
## 8  ( 1 ) "*"                   "*"
##          track_acousticness_IQR track_instrumentalness_mean
## 1  ( 1 ) " "                    " "
## 2  ( 1 ) " "                    " "
## 3  ( 1 ) " "                    " "
```

```
## 4  ( 1 ) " "                    " "
## 5  ( 1 ) "*"                    " "
## 6  ( 1 ) "*"                    " "
## 7  ( 1 ) "*"                    " "
## 8  ( 1 ) "*"                    " "
##          track_instrumentalness_IQR track_valence_mean track_valence_IQR
## 1  ( 1 ) " "                        " "                " "
## 2  ( 1 ) " "                        "*"                " "
## 3  ( 1 ) " "                        "*"                "*"
## 4  ( 1 ) " "                        "*"                "*"
## 5  ( 1 ) " "                        "*"                "*"
## 6  ( 1 ) " "                        "*"                "*"
## 7  ( 1 ) " "                        "*"                "*"
## 8  ( 1 ) " "                        "*"                "*"
##          track_tempo_mean track_tempo_IQR
## 1  ( 1 ) " "              " "
## 2  ( 1 ) " "              " "
## 3  ( 1 ) " "              " "
## 4  ( 1 ) " "              " "
## 5  ( 1 ) " "              " "
## 6  ( 1 ) "*"              " "
## 7  ( 1 ) "*"              " "
## 8  ( 1 ) "*"              " "
```

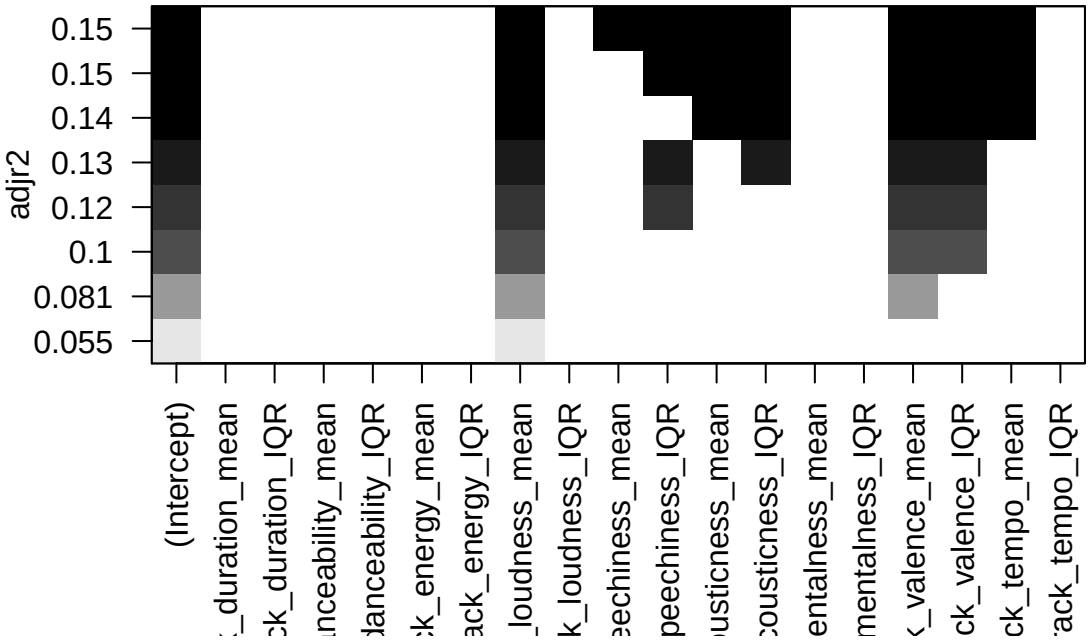
```r
#measures
par(mfrow = c(1,1))

plot(m, scale = "Cp")
```




```
plot(m, scale = "bic")
```




```
plot(m, scale = "adjr2")
```




```
#check Adjusted R2
rsq <- as.data.frame(summary(m)$adjr2)
names(rsq) <- "Adjusted-R2"
rsq %>%
        ggvis(x=~ c(1:nrow(rsq)), y=~`Adjusted-R2` ) %>%
        layer_points(fill = ~`Adjusted-R2`) %>%
        add_axis("y", title = "Adjusted-R2`") %>%
        add_axis("x", title = "Number of variables")
```

Renderer: SVG | Canvas

Download

```
#the final Adjusted R2 is very high

reg.summary <- summary(m)


#compare RSS, Cp, bic, adjr2
par(mfrow=c(2,2))
plot(reg.summary$rss ,xlab="Number of Variables ",ylab="RSS",type="l")
plot(reg.summary$adjr2 ,xlab="Number of Variables ", ylab="Adjusted RSq",type="l")
which.max(reg.summary$adjr2)
```

```
## [1] 8
```

```
points(which.max(reg.summary$adjr2),reg.summary$adjr2[which.max(reg.summary$adjr2)], col="red",cex=2,pch
plot(reg.summary$cp ,xlab="Number of Variables ",ylab="Cp", type='l')
```

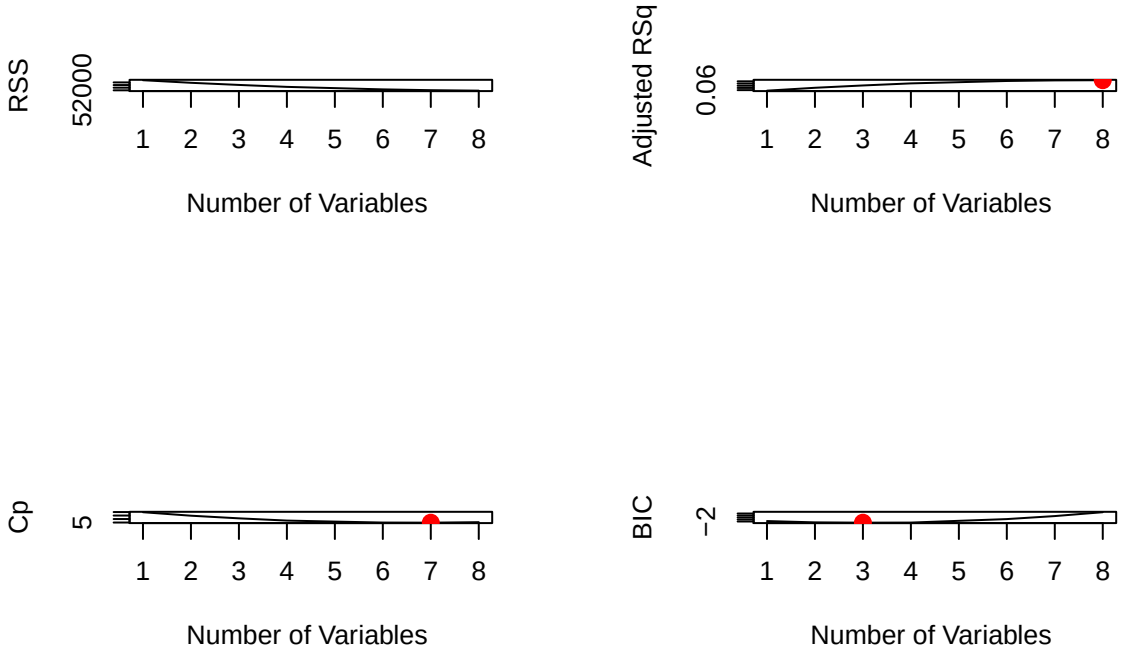
```
which.min(reg.summary$cp )
```

```
## [1] 7
```

```
points(which.min(reg.summary$cp ),reg.summary$cp [which.min(reg.summary$cp )],col="red",cex=2,pch=20)
plot(reg.summary$bic ,xlab="Number of Variables ",ylab="BIC",type='l')
which.min(reg.summary$bic )
```

```
## [1] 3
```

```
points(which.min(reg.summary$bic ),reg.summary$bic [which.min(reg.summary$bic )],col="red",cex=2,pch=20)
```




```
par(mfrow=c(1,1))


coef(m, 8)
```

```
##             (Intercept)     track_loudness_mean  track_speechiness_mean
##               86.1397393                1.4856365             -46.3035600
##   track_speechiness_IQR track_acousticness_mean  track_acousticness_IQR
##               74.2933223             -12.5405480              14.7223213
##      track_valence_mean       track_valence_IQR       track_tempo_mean
##             -19.6008770               21.3023592              -0.2415466
```

```
#a selected best model by adjr2
m3 <- lm( popularity ~ track_duration_mean +   track_loudness_mean + track_speechiness_mean + track_spe
summary(m3)
```

```
##
```

```
## Call:
## lm(formula = popularity ~ track_duration_mean + track_loudness_mean +
##     track_speechiness_mean + track_speechiness_IQR + track_acousticness_IQR +
##     track_valence_IQR + track_tempo_mean + track_tempo_IQR, data = data_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.170 -14.888  -0.371  13.984  43.740
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             8.033e+01  1.983e+01   4.051 7.46e-05 ***
## track_duration_mean     5.419e-06  1.310e-05   0.414   0.6795
## track_loudness_mean     1.851e+00  4.230e-01   4.375 2.01e-05 ***
## track_speechiness_mean -6.655e+01  4.510e+01  -1.476   0.1417
## track_speechiness_IQR   1.088e+02  4.930e+01   2.208   0.0285 *
## track_acousticness_IQR  1.648e+01  8.106e+00   2.033   0.0435 *
## track_valence_IQR       7.797e+00  1.238e+01   0.630   0.5297
## track_tempo_mean       -2.460e-01  1.351e-01  -1.821   0.0703 .
## track_tempo_IQR         2.972e-02  8.067e-02   0.368   0.7130
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.91 on 188 degrees of freedom
## Multiple R-squared:  0.1349, Adjusted R-squared:  0.09806
## F-statistic: 3.664 on 8 and 188 DF,  p-value: 0.0005423
```

```
#predictions on testing data
preds <- predict(m3, newdata = data_3 )

absolute_bs3 <- sqrt(mean((preds - data_3$popularity)^2))
mse_bs3 <- mean((preds - data_3$popularity)^2)
```

Looking at ridge regression... The commented lines were used to visualise the datasets in our dummy runs but were not rerun as part of the actual methods

```
#Get training_data_pop from EDA file
#Code based from http://www.science.smith.edu/~jcrouser/SDS293/labs/lab10-r.html
ridge_reg <- function(data){
model_data <- data
output <- data.frame(matrix(0, nrow = 0, ncol = 2))
#Looking at the models with the whole dataset
x <- model.matrix(popularity ~ ., model_data )
y <- define_y(model_data)

grid <- 10^seq(10, -2, length = 100)
ridge_mod <- glmnet(x, y, alpha = 0)

#Looking at the model
#dim(coef(ridge_mod))
#plot(ridge_mod)    # Draw plot of coefficients

output <- data.frame(matrix(0, nrow = 0, ncol = 2))
for(i in 1:nrow(model_data)){
  train = model_data[-i,]
```

```
test <- model_data %>%
  setdiff(train)

x_train <- model.matrix(popularity ~ ., train)
x_test <- model.matrix(popularity ~ ., test)

y_train <- define_y(train)


y_test <- define_y(test)

cv.out <- cv.glmnet(x_train, y_train, alpha = 0) # Fit ridge regression model on training data
bestlam <- cv.out$lambda.min  # Select lamda that minimizes training MSE
ridge_mod <- glmnet(x_train, y_train, alpha= 0, lambda = bestlam, thresh = 1e-12)
ridge_pred <- predict(ridge_mod, s = bestlam, newx = x_test) # Use best lambda to predict test data
m <- mean((ridge_pred - y_test)^2) # Calculate test MSE
output[i,1] <- bestlam
output[i,2] <- m
}
output <- output %>%
  rename(best_lambda = X1, MSE = X2)
#plot(cv.out)
mean_lambda <- output %>%
  #Using median rather than mean as since we are only using 1 datapoint
  #for our test dataset, if it is anonamous it could lead to a slight bias
  summarise(lambda = median(output$best_lambda)) %>%
  as_vector()
#Fitting the model on the whole dataset as defined by the average
#best lamda as calculated above using k-1 validation
#This is then returned as the output of the function
glmnet(x, y, alpha = 0, lambda = mean_lambda)
}
```

Running Ridge Regression on our three datasets and looking at MSE.

```
#Dataset 1
ridge_data1 <- ridge_reg(data_1)
#MSE of entire training dataset
x_data1 <- model.matrix(popularity ~ ., data_1)
y_data1 <- define_y(data_1)
ridge_pred_1 <- predict(ridge_data1, newx = x_data1)
mse_model1 <- mean((ridge_pred_1 - y_data1)^2)
absolute_ridge1 <- median(sqrt((ridge_pred_1 - y_data1)^2))
#Extract the coefficients of each variable in the model

#Dataset2
ridge_data2 <- ridge_reg(data_2)
#MSE of entire training dataset
x_data2 <- model.matrix(popularity ~ ., data_2)
y_data2 <- define_y(data_2)
ridge_pred_2 <- predict(ridge_data2, newx = x_data2)
mse_model2 <- mean((ridge_pred_2 - y_data2)^2)
absolute_ridge2 <- median(sqrt((ridge_pred_2 - y_data2)^2))
#Extract the coefficients of each variable in the model
```

```
#Dataset 3
ridge_data3 <- ridge_reg(data_3)
#MSE of entire training dataset
x_data3 <- model.matrix(popularity ~ ., data_3)
y_data3 <- define_y(data_3)
ridge_pred_3 <- predict(ridge_data3, newx = x_data3)
mse_model_3 <- mean((ridge_pred_3 - y_data3)^2)
absolute_ridge3 <- median(sqrt((ridge_pred_3 - y_data3)^2))
#Extract the coefficients of each variable in the model
```

Looking at the Lasso linear modelling method to see if we can improve on ridge regression

```
set.seed(666)
#Get training_data_pop from EDA file
#Code based from http://www.science.smith.edu/~jcrouser/SDS293/labs/lab10-r.html
lasso_reg <- function(data){
model_data <- data
output <- data.frame(matrix(0, nrow = 0, ncol = 2))
#Looking at the models with the whole dataset
x <- model.matrix(popularity ~ ., model_data )
y <- define_y(model_data)

grid <- 10^seq(10, -2, length = 100)
ridge_mod <- glmnet(x, y, alpha = 1)

output <- data.frame(matrix(0, nrow = 0, ncol = 2))
for(i in 1:nrow(model_data)){
  train = model_data[-i,]

test <- model_data %>%
  setdiff(train)

x_train <- model.matrix(popularity ~ ., train)
x_test <- model.matrix(popularity ~ ., test)

y_train <- define_y(train)


y_test <- define_y(test)

cv.out <- cv.glmnet(x_train, y_train, alpha = 1) # Fit lasso regression model on training data
bestlam <- cv.out$lambda.min  # Select lamda that minimizes training MSE
ridge_mod <- glmnet(x_train, y_train, alpha= 1, lambda = bestlam, thresh = 1e-12)
ridge_pred <- predict(ridge_mod, s = bestlam, newx = x_test) # Use best lambda to predict test data
m <- mean((ridge_pred - y_test)^2) # Calculate test MSE
output[i,1] <- bestlam
output[i,2] <- m
}
output <- output %>%
  rename(best_lambda = X1, MSE = X2)
#plot(cv.out)
mean_lambda <- output %>%
  #Using median rather than mean as since we are only using 1 datapoint
  #for our test dataset, if it is anonamous it could lead to a slight bias
```

```
  summarise(lambda = median(output$best_lambda)) %>%
  as_vector()
#Fitting the model on the whole dataset as defined by the average
#best lamda as calculated above using k-1 validation
#This is then returned as the output of the function
glmnet(x, y, alpha = 1, lambda = mean_lambda)
}
```

Lasso outputs

```
lasso_data1 <- lasso_reg(data_1)
lx_data1 <- model.matrix(popularity ~ ., data_1)
ly_data1 <- define_y(data_1)
lasso_pred_1 <- predict(lasso_data1, newx = lx_data1)
lmse_model1 <- mean((lasso_pred_1 - ly_data1)^2)
absolute_lasso1 <- median(sqrt((lasso_pred_1 - ly_data1)^2))
#Extract the coefficients of each variable in the model

lasso_data2 <- lasso_reg(data_2)
lx_data2 <- model.matrix(popularity ~ ., data_2)
ly_data2 <- define_y(data_2)
lasso_pred_2 <- predict(lasso_data1, newx = lx_data2)
lmse_model2 <- mean((lasso_pred_2 - ly_data2)^2)
absolute_lasso2 <- median(sqrt((lasso_pred_2 - ly_data2)^2))
#Extract the coefficients of each variable in the model

lasso_data3 <- lasso_reg(data_3)
lx_data3 <- model.matrix(popularity ~ ., data_3)
ly_data3 <- define_y(data_3)
lasso_pred_3 <- predict(lasso_data3, newx = lx_data3)
lmse_model3 <- mean((lasso_pred_3 - ly_data3)^2)
absolute_lasso3 <- median(sqrt((lasso_pred_3 - ly_data3)^2))
#Extract the coefficients of each variable in the model
```

From this it seems the ridge regression model performs better in terms of MSE although the Lasso Model has the benefit of discarding some paremeters as the values associated with them are zero, whereas ridge gives a value to every input (no subset selection). Summary of all the models:

```
model_summary <- tibble(method = c("Best Subset Selection 1", "Best Subset Selection 2", "Best Subset S
                   "Ridge 1","Ridge 2","Ridge 3", "Lasso 1", "Lasso 2", "Lasso 3"),
  MSE = c(mse_bs1, mse_bs2, mse_bs3, mse_model1, mse_model2, mse_model_3, lmse_model1, lmse_model2, lms
       AE = c(absolute_bs1, absolute_bs2, absolute_bs3, absolute_ridge1, absolute_ridge2,
              absolute_ridge3, absolute_lasso1, absolute_lasso2, absolute_lasso3))
model_summary
```

```
## # A tibble: 9 x 3
##   method                     MSE    AE
##   <chr>                    <dbl> <dbl>
## 1 Best Subset Selection 1  279.  16.7
## 2 Best Subset Selection 2  328.  18.1
## 3 Best Subset Selection 3  286.  16.9
## 4 Ridge 1                  277.  12.3
## 5 Ridge 2                  325.  14.1
## 6 Ridge 3                  266.  10.7
## 7 Lasso 1                  281.  12.4
```

```
## 8 Lasso 2                  332.  14.7
## 9 Lasso 3                  270.  11.1
```

```
ridge_best <- ridge_data3$beta
ridge_best
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)                  .
## track_duration_mean         -1.762525e-06
## track_duration_IQR           7.572830e-06
## track_danceability_mean      3.860313e+00
## track_danceability_IQR       9.576059e+00
## track_energy_mean            6.770943e+00
## track_energy_IQR            -1.478744e+00
## track_loudness_mean          6.817506e-01
## track_loudness_IQR          -3.885396e-01
## track_speechiness_mean      -9.986200e-01
## track_speechiness_IQR        2.080335e+01
## track_acousticness_mean     -7.789401e+00
## track_acousticness_IQR       1.051796e+01
## track_instrumentalness_mean -4.743061e+00
## track_instrumentalness_IQR   4.962972e+00
## track_valence_mean          -1.355098e+01
## track_valence_IQR            1.205485e+01
## track_tempo_mean            -1.457377e-01
## track_tempo_IQR              5.766492e-02
```

```
lasso_best <- lasso_data3$beta
lasso_best
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)                  .
## track_duration_mean          .
## track_duration_IQR           .
## track_danceability_mean      .
## track_danceability_IQR       .
## track_energy_mean            .
## track_energy_IQR             .
## track_loudness_mean          1.05245294
## track_loudness_IQR           .
## track_speechiness_mean       .
## track_speechiness_IQR       20.44239244
## track_acousticness_mean     -8.63975604
## track_acousticness_IQR       9.61726094
## track_instrumentalness_mean  .
## track_instrumentalness_IQR   .
## track_valence_mean          -13.60226769
## track_valence_IQR            13.08127761
## track_tempo_mean            -0.12010942
## track_tempo_IQR              0.02317519
```

To save time, if the output needs to be seen quickly then the following rcode loads a previously saved image of the R environment so all the variables can be referenced. Commented out as only needs to be run if not knitting and want to see variables quickly

```
#saving the rdata
#save.image(file = "models.Rdata")
#Can be loaded with:
#load("models.Rdata)
```