R Notebook

```
{r, warning = FALSE} library(readr) library(tidyverse) library(dplyr)
library(leaps) library(car) library(MASS)
```

```
{r, echo = FALSE} df_spotify <- read_csv("spotify-2023.csv")
#glimpse(df_spotify) #names(df_spotify)
```

1. Summary Statistics of variables selected for MLR

```
# Streams was originally of chr type. Changed it to dbl type.
df_spotify <- df_spotify %>% mutate(streams = as.numeric(streams))

par(mfrow=c(3,3))
hist(df_spotify$`danceability_%`)
hist(df_spotify$bpm)
hist(df_spotify$streams)
hist(df_spotify$`speechiness_%`)
hist(df_spotify$`valence_%`)
hist(df_spotify$`energy_%`)
hist(df_spotify$`acousticness_%`)
hist(df_spotify$`liveness_%`)
hist(df_spotify$`instrumentalness_%`)
# Remove this from OG model - data points don't make sense it context
```

```
# List your predictor columns exactly as they appear in your data
predictor_cols <- c("danceability_%",
  "bpm", "streams", "valence_%", "energy_%", "acousticness_%", "liveness_%", "speechiness_%"
  "mode_binary", "in_spotify_playlists"
)

# Creating mode_binary variable which is 1 if mode is Major and 0 for Minor.
df_spotify <- df_spotify %>% mutate(mode_binary = ifelse(mode == 'Major', 1, 0))

df_pred <- df_spotify %>%
  mutate(
    streams = as.numeric(gsub(",", "", as.character(streams))),
    mode_binary = as.numeric(as.character(mode_binary))
  ) %>%
  select(all_of(predictor_cols))

summary_table <- map_dfr(names(df_pred), function(v) {
  x <- df_pred[[v]]
  tibble(
    variable = v,
    n   = sum(!is.na(x)),
    mean = round(mean(x, na.rm = TRUE), 2),
    sd   = round(sd(x, na.rm = TRUE), 2),
    min  = round(suppressWarnings(min(x, na.rm = TRUE)), 2),
    max  = round(suppressWarnings(max(x, na.rm = TRUE)), 2)
  )
})

summary_table
```

2. Data Cleaning for MLR

```
# Creating mode_binary variable which is 1 if mode is Major and 0 for Minor.
df_spotify <- df_spotify %>% mutate(mode_binary = ifelse(mode == 'Major', 1, 0))

# Streams was originally of chr type. Changed it to dbl type.
df_spotify <- df_spotify %>% mutate(streams = as.numeric(streams))

# Note that in streams, observation #575 had an inconsistency: instead of displaying # of st
df_spotify <- df_spotify %>% filter(track_name != "Love Grows (Where My Rosemary Goes)")
```

Justification for interaction term prior to fitting model

```
major <- df_spotify[df_spotify$mode_binary == 1, ]
minor <- df_spotify[df_spotify$mode_binary == 0, ]

# Note that danceability is response variable
par(mfrow=c(2,2))
boxplot(df_spotify$`danceability_%`~ df_spotify$mode_binary)

# plotting relationship between Danceability % and bpm for major/minor
plot(df_spotify$`danceability_%` ~ df_spotify$bpm, xlab="Bpm", ylab="Danceability %", type="
points(major$`danceability_%` ~ major$bpm, col="blue")
points(minor$`danceability_%` ~ minor$bpm, col="black")
lines(lowess(major$`danceability_%` ~ major$bpm), col="blue")
lines(lowess(minor$`danceability_%` ~ minor$bpm), col="black")

# plotting relationship between Danceability % and `valence_%` for major/minor
plot(df_spotify$`danceability_%` ~ df_spotify$`valence_%`, xlab="`valence_%`", ylab="Danceab
points(major$`danceability_%` ~ major$`valence_%`, col="blue")
points(minor$`danceability_%` ~ minor$`valence_%`, col="black")
```

```
lines(lowess(major$`danceability_%` ~ major$`valence_%`), col="blue")
lines(lowess(minor$`danceability_%` ~ minor$`valence_%`), col="black")

# plotting relationship between Danceability % and `speechiness_%`s for major/minor
plot(df_spotify$`danceability_%` ~ df_spotify$`speechiness_%`, xlab="``speechiness_%`", ylab
points(major$`danceability_%` ~ major$`speechiness_%`, col="blue")
points(minor$`danceability_%` ~ minor$`speechiness_%`, col="black")
lines(lowess(major$`danceability_%` ~ major$`speechiness_%`), col="blue")
lines(lowess(minor$`danceability_%` ~ minor$`speechiness_%`), col="black")

# Since relationship between danceability % and speechiness % seems to differ for major/mino
```

2. Fit MLR Model

```
# Ensures categorical use for mode
df_spotify$mode_binary <- factor(df_spotify$mode_binary)

model <- lm(`danceability_%` ~ bpm  + streams + `valence_%` + `energy_%` + `acousticness_%`

summary(model)
```

3. Checking MLR Assumptions

```
# Need to check for linearity, constant variance, uncorrelated errors, and normality.
# How to? Plots!

# Residuals vs fitted plot
y_value <- resid(model)
x_value <- fitted(model)
plot(x = x_value, y = y_value, main="Residual vs Fitted", xlab="Fitted", ylab="Residuals")

par(mfrow=c(1,2))
# QQ Normality plots
qqnorm(resid(model), main = "Normal Q-Q Plot")
qqline(resid(model))


# Response vs Fitted plot
y_value2 <- df_spotify$bpm
x_value2 <- fitted(model)
plot(x = x_value2, y = y_value2, main="Response vs Fitted", xlab="Fitted", ylab="Actual Resp


# Plot of relationship between predictor variables and also outcome variable
pairs(c(df_spotify[, 18], df_spotify[, 7], df_spotify[, 9], df_spotify[, 15], df_spotify[, 1
par(mfrow=c(2,2))
# Residuals vs each predictor
plot(x = df_spotify$bpm, y = y_value, main="Residual vs Bpm", xlab="Bpm", ylab="Residual")
plot(x = df_spotify$streams, y = y_value, main="Residual vs Streams", xlab="Streams", ylab="
plot(x = df_spotify$`valence_%`, y = y_value, main="Residual vs Valence %", xlab="Valence %"
plot(x = df_spotify$`energy_%`, y = y_value, main="Residual vs Energy %", xlab="Energy %", y

par(mfrow=c(2,2))
plot(x = df_spotify$`acousticness_%`, y = y_value, main="Residual vs Acousticness %", xlab="
plot(x = df_spotify$`liveness_%`, y = y_value, main="Residual vs Liveness %", xlab="Liveness
plot(x = df_spotify$`speechiness_%`, y = y_value, main="Residual vs Speechiness %", xlab="Sp

par(mfrow=c(1,2))
plot(x = df_spotify$in_spotify_playlists, y = y_value, main="Residual vs in_spotify_playlist
# For categorical variable
boxplot(y_value ~ df_spotify$mode_binary , main="Residual vs Mode", xlab="Mode", ylab="Resic
```

HOW TO FIX:

1. Check Multi-collinearity among predictors
2. Check points that simply don't make sense from descriptive stats
3. Transformations
4. Nested models

#1. Multi collinearity

```
# x includes all predictors in original model: in_spotify_playlists, streams, bpm, valence_%
x <- cbind(df_spotify[, 7], df_spotify[, 9], df_spotify[, 15], df_spotify[, 19:21], df_spoti

x_num <- as.matrix(x)

qr(x_num)$rank
ncol(x_num)
# This shows that the rank is the same as number of columns (no lin dependent columns). FULL

# vif(model)
# No vif > 5 = keep all predictors.

x_numeric <- x %>% select(-mode_binary)
cor(x_numeric)
# Also supports previous points
```

```
# THUS, NO MULTICOLLINEARITY IN OG MODEL
```

#2. Check for problematic observations

```
# leverage statistic
h_jj <- hatvalues(model)
# standardized residuals
r_j <- rstandard(model)
# cook's distance
D_j <- cooks.distance(model)
# dffits
dffits_j <- dffits(model)
# dfbetas
dfbetas_j <- dfbetas(model)

# number of predictors and observations
n <- nobs(model)
p <- length(coef(model)) - 1

# leverage cutoff
hcut <- 2 * ((p +1) / n)
# cook's distance cutoff
cookcut <- qf(0.5, df1 = p + 1, df2 = n - p - 1)
# dffits cutoff
fitcut <- 2 * sqrt((p+1)/n)
# dfbeta cutoff
betacut <- 2 / (sqrt(n))

# which observations are leverage points?
which(h_jj > hcut)
# which observations are regression outliers?
which(r_j > 4 | r_j < -4)
# which observations are influential by cook's distance?
which(D_j > cookcut)
# which observations are influential by dffits?
which(abs(dffits_j) > fitcut)

# After initial check of problematic observations, we realized that high influential points

# NO RATIONALE TO REMOVE ANY SONGS.
```

#3. Transformation of Variables

Normality

```
p_box <- powerTransform(cbind(df_spotify[, 18], df_spotify[, 7], df_spotify[, 9], df_spotify
#summary(p_box)

# fix worst transformation first, check residual plots after every transformation, avoid too

# 1. Normality
```

```
t_y <- (df_spotify$`danceability_%`) ^ 2 # value of lambda obtained by BoxCox

model_1 <- lm(t_y ~ bpm  + streams + `valence_%` + `energy_%` + `acousticness_%` + `liveness
```

```
# Now we check residual plots

par(mfrow=c(2,2))
y_value_1 <- resid(model_1)
x_value_1 <- fitted(model_1)
plot(x = x_value_1, y = y_value_1, main="Residual vs Fitted Model 1", xlab="Fitted", ylab="R

# Residuals vs each predictor
plot(x = df_spotify$bpm, y = y_value_1, main="Residual vs Bpm", xlab="Bpm", ylab="Residual")
plot(x = df_spotify$streams, y = y_value_1, main="Residual vs Streams", xlab="Streams", ylab
plot(x = df_spotify$`valence_%`, y = y_value_1, main="Residual vs Valence %", xlab="Valence 

par(mfrow=c(2,2))
plot(x = df_spotify$`energy_%`, y = y_value_1, main="Residual vs Energy %", xlab="Energy %",

plot(x = df_spotify$`acousticness_%`, y = y_value_1, main="Residual vs Acousticness %", xlab
plot(x = df_spotify$`liveness_%`, y = y_value_1, main="Residual vs Liveness %", xlab="Livene
plot(x = df_spotify$`speechiness_%`, y = y_value_1, main="Residual vs Speechiness %", xlab="

par(mfrow=c(2,2))
plot(x = df_spotify$in_spotify_playlists, y = y_value_1, main="Residual vs in_spotify_playli
# For categorical variable
boxplot(y_value ~ df_spotify$mode_binary , main="Residual vs Mode", xlab="Mode", ylab="Resid

# Check normality just in case
# QQ Normality plots
qqnorm(resid(model_1), main = "Normal Q-Q Plot")
qqline(resid(model_1))

# random noise at tails
```

Constant Variance

In_spotify_playlist

```
# From Residual plots, we see that multiple predictors exhibit non constant variance. First,
```

```
t_playlist <- log(df_spotify$in_spotify_playlists)

model_2 <- lm(t_y ~ bpm  + streams + `valence_%` + `energy_%` + `acousticness_%` + `liveness
```

```
# Now we check for residuals
```

```
par(mfrow=c(2,2))
y_value_1 <- resid(model_2)
x_value_1 <- fitted(model_2)
plot(x = x_value_1, y = y_value_1, main="Residual vs Fitted Model 1", xlab="Fitted", ylab="

# Residuals vs each predictor
plot(x = df_spotify$bpm, y = y_value_1, main="Residual vs Bpm", xlab="Bpm", ylab="Residual")
plot(x = df_spotify$streams, y = y_value_1, main="Residual vs Streams", xlab="Streams", ylab
plot(x = df_spotify$`valence_%`, y = y_value_1, main="Residual vs Valence %", xlab="Valence

par(mfrow=c(2,2))
plot(x = df_spotify$`energy_%`, y = y_value_1, main="Residual vs Energy %", xlab="Energy %",

plot(x = df_spotify$`acousticness_%`, y = y_value_1, main="Residual vs Acousticness %", xlab
plot(x = df_spotify$`liveness_%`, y = y_value_1, main="Residual vs Liveness %", xlab="Livene
plot(x = df_spotify$`speechiness_%`, y = y_value_1, main="Residual vs Speechiness %", xlab="

par(mfrow=c(2,2))
plot(x = t_playlist, y = y_value_1, main="Residual vs in_spotify_playlists", xlab="in_spotif
# For categorical variable
boxplot(y_value ~ df_spotify$mode_binary , main="Residual vs Mode", xlab="Mode", ylab="Resid

# Check normality just in case
# QQ Normality plots
qqnorm(resid(model_2), main = "Normal Q-Q Plot")
qqline(resid(model_2))
```

Streams

```
t_streams <- log(df_spotify$streams) # boxcox said 0.14 and log looks like cluster so better

model_3 <- lm(t_y ~ bpm  + t_streams + `valence_%` + `energy_%` + `acousticness_%` + `livene
```

```
# Now we check for residuals

par(mfrow=c(2,2))
y_value_1 <- resid(model_3)
x_value_1 <- fitted(model_3)
plot(x = x_value_1, y = y_value_1, main="Residual vs Fitted Model 1", xlab="Fitted", ylab="

# Residuals vs each predictor
plot(x = df_spotify$bpm, y = y_value_1, main="Residual vs Bpm", xlab="Bpm", ylab="Residual")
plot(x = t_streams, y = y_value_1, main="Residual vs Streams", xlab="Streams", ylab="Residua
plot(x = df_spotify$`valence_%`, y = y_value_1, main="Residual vs Valence %", xlab="Valence

par(mfrow=c(2,2))
plot(x = df_spotify$`energy_%`, y = y_value_1, main="Residual vs Energy %", xlab="Energy %",
plot(x = df_spotify$`acousticness_%`, y = y_value_1, main="Residual vs Acousticness %", xlab
plot(x = df_spotify$`liveness_%`, y = y_value_1, main="Residual vs Liveness %", xlab="Livene
plot(x = df_spotify$`speechiness_%`, y = y_value_1, main="Residual vs Speechiness %", xlab="

par(mfrow=c(2,2))
plot(x = t_playlist, y = y_value_1, main="Residual vs in_spotify_playlists", xlab="in_spotif
# For categorical variable
boxplot(y_value ~ df_spotify$mode_binary , main="Residual vs Mode", xlab="Mode", ylab="Resid

# Check normality just in case
# QQ Normality plots
qqnorm(resid(model_3), main = "Normal Q-Q Plot")
qqline(resid(model_3))
```

Recall from week 9 - BEST model selection

```
best <- regsubsets(t_y ~ bpm  + t_streams + `valence_%` + `energy_%` + `acousticness_%` + `l

summary(best)
```

#4. Nested Models

```
summary(model_3) # By looking at summary, we see that t_streams,  and interaction term are n

# Thus, we build reduced model without those
# Further supported by best selection model

model_3_reduced <- lm(t_y ~ bpm + t_playlist  + `valence_%` + `energy_%` + `acousticness_%`

anova(model_3_reduced, model_3)
# small p value = drop = full model better
# large p value = reduced model better

summary(model_3_reduced)
```

```
# Now we check for residuals

par(mfrow=c(2,2))
y_value_1 <- resid(model_3_reduced)
x_value_1 <- fitted(model_3_reduced)
plot(x = x_value_1, y = y_value_1, main="Residual vs Fitted Model 1", xlab="Fitted", ylab="R

# Residuals vs each predictor
plot(x = df_spotify$bpm, y = y_value_1, main="Residual vs Bpm", xlab="Bpm", ylab="Residual")
plot(x = t_streams, y = y_value_1, main="Residual vs Streams", xlab="Streams", ylab="Residua
plot(x = df_spotify$`valence_%`, y = y_value_1, main="Residual vs Valence %", xlab="Valence

par(mfrow=c(2,2))
plot(x = df_spotify$`energy_%`, y = y_value_1, main="Residual vs Energy %", xlab="Energy %",

plot(x = df_spotify$`acousticness_%`, y = y_value_1, main="Residual vs Acousticness %", xlab
plot(x = df_spotify$`liveness_%`, y = y_value_1, main="Residual vs Liveness %", xlab="Livene
plot(x = df_spotify$`speechiness_%`, y = y_value_1, main="Residual vs Speechiness %", xlab="

par(mfrow=c(2,2))
plot(x = t_playlist, y = y_value_1, main="Residual vs in_spotify_playlists", xlab="in_spotif
# For categorical variable
boxplot(y_value ~ df_spotify$mode_binary , main="Residual vs Mode", xlab="Mode", ylab="Resic

# Check normality just in case
# QQ Normality plots
qqnorm(resid(model_3_reduced), main = "Normal Q-Q Plot")
qqline(resid(model_3_reduced))
```

Model Selection

```
# Full transformed model (same as  model_3)
full_model <- lm(
  t_y ~ bpm + t_streams + `valence_%` + `energy_%` +
         `acousticness_%` + `liveness_%` + `speechiness_%` +
         mode_binary + t_playlist + mode_binary*`speechiness_%`,
  df_spotify
)

# Null model, intercept only
null_model <- lm(t_y ~ 1, data = df_spotify)

# Forward Selection with AIC
forward_model <- stepAIC(
  null_model,
  scope = list(
    lower = null_model,
    upper = full_model
  ),
  direction = "forward",
  trace = TRUE
)

summary(forward_model)
```

```
# Backward

backward_model <- stepAIC(
  full_model,
  direction = "backward",
  trace = TRUE
)

summary(backward_model)
```

```
# Stepwise

stepwise_model <- stepAIC(
  full_model,
  direction = "both",
  trace = TRUE
)

summary(stepwise_model)
```

```
AIC(full_model, forward_model, backward_model, stepwise_model)

BIC(full_model, forward_model, backward_model, stepwise_model)

c(
  full = summary(full_model)$adj.r.squared,
  forward = summary(forward_model)$adj.r.squared,
  backward = summary(backward_model)$adj.r.squared,
  stepwise = summary(stepwise_model)$adj.r.squared
)


# all selection algorithms give same model as each other and as the anova test. and has bett
```