Dataset Sourcing and Application Description

The Free Music Archive (FMA) dataset (Defferrard et al. (2016)) is a public collection of music tracks and metadata designed for music information retrieval research. Sourced from the Free Music Archive platform, it contains over 106,000 musical tracks along with a variety of features, including track titles, artist names, genre classifications, and listening statistics. It also includes detailed audio features extracted by the Echonest (Ellis et al. (2010)), a music intelligence platform acquired by Spotify, which uses advanced algorithms to analyze tracks. These 8 audio features—acousticness, danceability, energy, instrumentalness, liveness, speechiness, tempo, and valence—are widely used for tasks such as genre classification and music recommendation. Each observation in the dataset represents an individual music track with rich metadata and audio characteristics, making it useful for genre classification and other forms of music analysis. For this study, a subset of 4340 tracks was selected, focusing on those with available Echonest features.

This study investigates whether a model can accurately classify music tracks into the top two genres—Electronic and Rock—based on the 8 Echonest audio features extracted from the FMA dataset. Specifically, a classification task where logistic regression and KNN algorithms will be used to predict the genre of a track. The goal is to determine the model's effectiveness in predicting genre classifications and compare the performance of these algorithms.

Exploratory Data Analysis

All data processing, analyses, and visualization was performed in R (R Core Team (2020)). The tidy dataset was saved in an R data format. During data preparation, missing values in the `track_genre_top` column were removed and only tracks labeled as *Electronic* or *Rock*—the two genres with the largest sample sizes—were selected for analysis. Our final dataset includes 2,170 tracks from the Electronic genre and 2,170 tracks from the Rock genre, resulting in a total of 4,340 samples. The dataset consists of both numerical and categorical data. The 8 Echonest features (predictor variables) are continuous numerical values, while `track_genre_num` is a binary categorical variable representing the genre. All predictor variables, with the exception of tempo, originally range from 0 to 1. To ensure consistency across the features, the tempo variable was scaled to a 0-1 range.

A correlation matrix (see Figure 1 in Supplementary) shows the relationships between the 8 Echonest features and the genre label (track_genre_num). Several features show notable correlations with the genre, such as danceability (-0.45) and acousticness (0.22), suggesting that these features may play a significant role in distinguishing between Electronic and Rock genres. The matrix also highlights relationships between features themselves.

Model Training and Evaluation

The dataset was split into 70% training and 30% hold-out sets to have a sufficiently large training set for model development, and preserving a separate test set to assess generalization performance of the model while ensuring that both genres are evenly represented in both sets. A 70-30 split is common in classification tasks as it gives sufficient data for training the models while keeping a reasonable amount of data for evaluating the model's performance on unseen data.

All 8 Echonest features were used as predictors. The K-value was tuned for KNN. A range of k=1 to 25 was tested using 5-fold cross-validation to identify the optimal k-value (see Figure 2 in Supplementary). The cross-validation process helps in evaluating different k-values to find the one that minimizes the classification error on the validation folds. After splitting the data, the KNN model with k=7 was applied to the test set. Although k=16 had the lowest error rate, values after k=7 had diminishing returns in error reduction, making k=7 a more balanced choice between model simplicity and performance.

The KNN gave an accuracy of 77.3%. The KNN model's sensitivity is 73.7% and specificity is 81.0%. The misclassification error rate is 22.7%, meaning about 22.7% of the tracks were incorrectly classified. The logistic regression model was also applied to the test set, with predictions made using a cutoff of 0.5. The logistic regression model had a sensitivity of 77.6% and a specificity of 77.0%. The misclassification error rate for the logistic regression model is 22.4%, which is slightly lower than KNN, showing that logistic regression misclassifies fewer tracks overall.

Logistic Regression with Shrinkage

The shrinkage value that performed best on this dataset was a lambda value of 0.03296, which is the largest value of lambda within one standard error of the minimum cross-validation error. This value was chosen for its balanced regularization and predictive performance, reducing over-fitting.

Without shrinkage, all predictor variables are included in the logistic regression model, even if they contribute minimally, which may result in over-fitting. With LASSO shrinkage, variables such as liveness and tempo were eliminated, allowing the model to focus on key predictors like danceability, acousticness, and valence, simplifying the model and improving its generalizability.

Without Shrinkage	With Shrinkage
Accuracy = 77.3%	Accuracy = 77.0%
Sensitivity = 77.6%	Sensitivity = 81.1%
Specificity = 77.0%	Specificity = 72.8%

When comparing the performance of the logistic regression model with and without shrinkage, accuracy is very similar. However, the sensitivity of the model increased significantly with shrinkage. The specificity of the model decreased slightly with shrinkage. The shrinkage model is preferable in this case because it simplifies the model by excluding irrelevant predictors. By shrinking coefficients for irrelevant predictors to zero, the LASSO model focuses on the most impactful features, improving generalizability. This helps avoid over-fitting and reduces model complexity while still achieving similar classification performance.

Generative AI Use

I asked ChatGPT (ChatGPT (2024)) question 2: to "identify an application (classification) that can be investigated with the FMA dataset". The response generated by ChatGPT suggested that I can use the FMA dataset to develop models to automatically classify songs into genres based on audio features, involving feature extraction (MFCCs, spectral contrast), machine learning models (decision trees, SVMs, neural networks), and evaluation (accuracy, precision, recall). It also gave me other potential applications including music recommendation systems, music similarity analysis, and artist classification. I think that generative AI tools could be helpful with deciding on how to split a dataset, since most ML models use higher ratios for train/test splits, so doing some initial exploration with AI-generated advice could provide a starting point and insights into this assignment.

References

ChatGPT. 2024. "Identify an Application (Classification) That Can Be Investigated with the FMA Dataset." https://chat.opeani.com.

Defferrard, Michaël, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. 2016. "FMA: A Dataset for Music Analysis." *arXiv Preprint arXiv:1612.01840*.

Ellis, Daniel PW, Brian Whitman, Tristan Jehan, and Paul Lamere. 2010. "The Echo Nest Musical Fingerprint."

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Supplementary Materials

```
# FMA Dataset
# https://github.com/mdeff/fma
# Pull Data
temp <- paste(tempfile(), ".zip", sep = "")
options(timeout = 60 * 10)
download.file("https://os.unil.cloud.switch.ch/fma/fma_metadata.zip", temp)

# Feature Data
# Consolidate multiline header.
echonest_colnames <- unz(temp, "fma_metadata/echonest.csv") %>%
  read_csv(n_max = 0, skip = 2) %>%
  rename(track_ID = "...1") %>%
  names()
# Read the data.
echonest_raw <- unz(temp, "fma_metadata/echonest.csv") %>%
  read_csv(skip = 4, col_names = echonest_colnames) %>%
  # Transform track_ID to integer for tibble merging
  mutate(track_ID = as.integer(track_ID))

# Remove temporal features
echonest <- echonest_raw[, c(1:26)]

# Metadata
# Consolidate multiline header.
metadata_colnames_a <- unz(temp, "fma_metadata/tracks.csv") %>%
  read_csv(n_max = 0, skip = 1) %>%
  rename(track_ID = `...1`) %>%
  names() %>%
  # Removing strange enncoding
  sub("\\...*", "", .)
```

```r
metadata_colnames_b <- unz(temp, "fma_metadata/tracks.csv") %>%
  read_csv(n_max = 0) %>%
  names() %>%
  # Removing strange enncoding
  sub("\\...*", "", .)
metadata_colnames <- paste(metadata_colnames_b, metadata_colnames_a, sep = "_")

# Read the data.
metadata_raw <- unz(temp, "fma_metadata/tracks.csv") %>%
  read_csv(skip = 3, col_names = metadata_colnames) %>%
  rename(track_ID = `_track_ID`) %>%
  # Transform track_ID to integer for tibble merging
  mutate(track_ID = as.integer(track_ID))
# Combine the data and metadata
data <- inner_join(metadata_raw, echonest, by = "track_ID")
# Clean up downloaded files
unlink(temp)

# Tidy Data
df_tidy <- data %>%
  select(c("track_genre_top", "track_listens", "acousticness", "danceability",
           "energy", "instrumentalness", "liveness", "speechiness", "tempo",
           "valence")) %>%
  # Dropping rows with missing genre
  drop_na(., track_genre_top) %>%
  # Selecting the two genres with the largest sample sizes
  filter(track_genre_top %in% c("Electronic", "Rock"))

# Write .RData
save(df_tidy, file = "df_tidy.RData")
```

```r
# Load data parsed above to minimize linting time.
load("df_tidy.RData")

# Add a numeric column for 'track_genre_top'
df_tidy <- df_tidy %>%
  mutate(track_genre_num = case_when(
    track_genre_top == "Electronic" ~ 0,
    track_genre_top == "Rock" ~ 1
  ),
  tempo = (tempo - min(tempo)) / (max(tempo) - min(tempo)))

# Compute the correlation matrix
cor_matrix <- cor(df_tidy %>%
                    select(-c("track_genre_top",
                              "track_listens")))
ggcorrplot(cor_matrix, lab = TRUE, lab_size = 2.5)+
  theme(axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10))
```

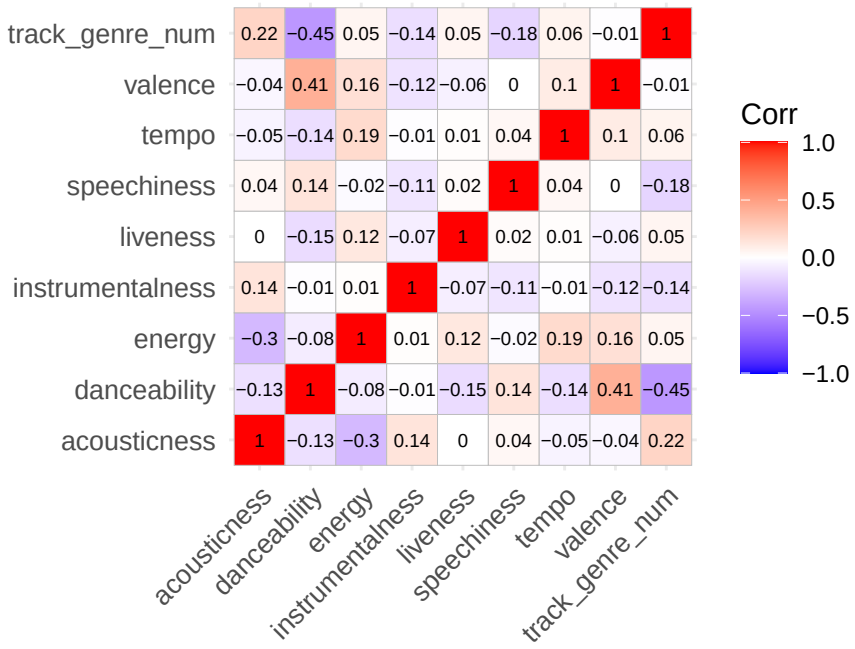

Figure 1: Correlation matrix between all 8 Echonest features and genre.

```
# Sample 2170 observations from each genre
sample <- df_tidy %>%
  group_by(track_genre_top) %>%
  sample_n(2170) %>%
  ungroup()
```

```
plot(genre_knn_cv, main = "KNN Cross-Validation Performance")
```

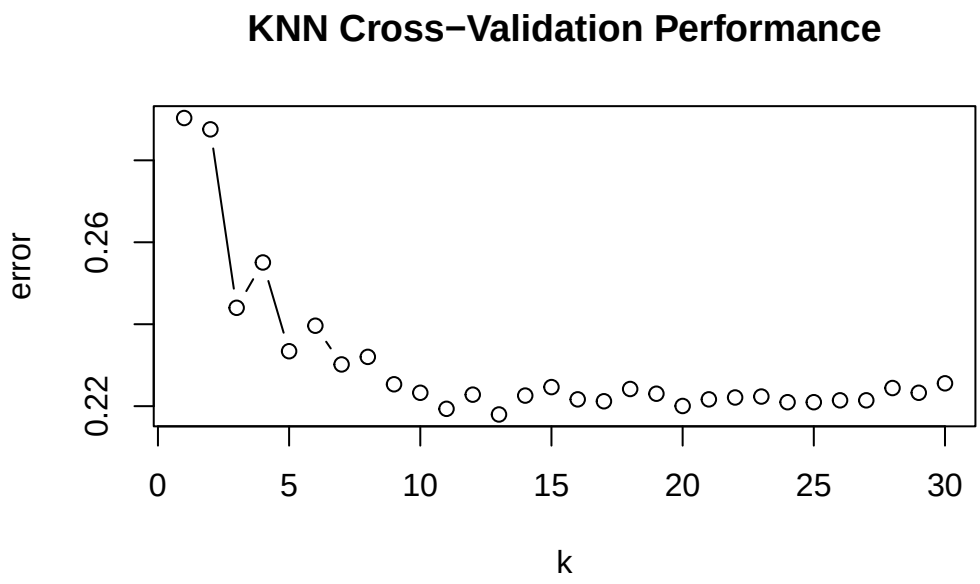

Figure 2: Results of k-fold cross validation for the knn model.

```
# KNN
# Predict genre based on variables
set.seed(123)
sample$track_genre_num <- as.factor(sample$track_genre_num)
index <- caret::createDataPartition(sample$track_genre_num,
                                    p = 0.7,
                                    list = FALSE)
genre_train <- slice(sample, index)
genre_test <- slice(sample, -index)

genre_x_train <- select(genre_train, -c(track_genre_top, track_genre_num, track_listens))
genre_y_train <- pull(genre_train, track_genre_num)

genre_x_test <- select(genre_test, -c(track_genre_top, track_genre_num, track_listens))
genre_y_test <- pull(genre_test, track_genre_num)

# Test best k value
```

```r
genre_x <- select(sample, -c(track_genre_top, track_genre_num, track_listens))
genre_y <- pull(sample, track_genre_num)

genre_knn_cv <- tune.knn(
  genre_x,
  genre_y,
  k = 1:25,
  tunecontrol = tune.control(
    sampling = "cross", cross = 5
  )
)
summary(genre_knn_cv)
plot(genre_knn_cv)

# Use K = 7
knn_pred <- knn(train = genre_x_train,
                test = genre_x_test,
                cl = genre_y_train,
                k = 7)

confusionMatrix(knn_pred, genre_y_test)

# Logistic Regression
logistic_model <- glm(track_genre_num ~ acousticness + danceability + energy +
                        instrumentalness + liveness + speechiness +
                        tempo + valence,
                      data = genre_train,
                      family = binomial)
summary(logistic_model)

pred_prob <- predict(logistic_model, newdata = genre_test, type = "response")
```

```r
genre_logreg_stat <- tibble(
  pre_prob = pred_prob,
  Y = genre_test$track_genre_num  # Actual test labels (0 = Electronic, 1 = Rock)
)

cutoff <- 0.5
genre_logreg_stat <- genre_logreg_stat %>%
  mutate(
    Class_predicted = ifelse(pre_prob > cutoff, 1, 0)
  )  # 1 = Rock, 0 = Electronic
tab <- table(genre_logreg_stat$Y, genre_logreg_stat$Class_predicted)

# miss-classification error rate
miss <- 1 - sum(diag(tab)) / sum(tab)

# sensitivity - % correctly predicting as default
sens <- tab[2, 2] / sum(tab[2, ])

# specificity - % correctly predicting as not default
spec <- tab[1, 1] / sum(tab[1, ])

# Logistic regression with shrinkage
X_train <- model.matrix(track_genre_num ~ acousticness + danceability + energy +
                                          instrumentalness + liveness + speechiness +
                                          tempo + valence,
                                          data = genre_train)[, -1]

Y_train <- dplyr::pull(genre_train, track_genre_num)

X_test <- model.matrix(track_genre_num ~ acousticness + danceability + energy +
```

```
                                    instrumentalness + liveness + speechiness +
                                    tempo + valence,
                                    data = genre_test)[, -1]
Y_test <- dplyr::pull(genre_test, track_genre_num)

cv_lasso <- cv.glmnet(
  X_train,
  Y_train,
  family = 'binomial',
  alpha = 1,
  standardize = FALSE
)

# Predict the log of odds of being 1 (Rock)
test_set_pred_log_odds <- predict(
  cv_lasso,
  newx = X_test,
  s = "lambda.1se"
)

# Convert log odds to probabilities
test_set_pred_prob <- exp(test_set_pred_log_odds) / (1 + exp(test_set_pred_log_odds))

# make a table with predicted prob and labels
df_lasso_test <- tibble(
  pre_prob = test_set_pred_prob,
  Y = Y_test # Actual test labels (0 = Electronic, 1 = Rock)
)
predict(
  cv_lasso,
  type = "coefficients",
```

```r
  s = cv_lasso$lambda.1se
  )

# Find cut-off based on ROC analysis
pROC_graph <- roc(
  df_lasso_test$Y,
  df_lasso_test$pre_prob,
  smoothed = TRUE,
  ci = TRUE,
  ci.alpha = 0.9,
  stratified = FALSE,
  plot = TRUE,
  auc.polygon = TRUE,
  max.auc.polygon = TRUE,
  grid = TRUE,
  print.auc = TRUE,
  show.thres = TRUE
)

coords_output <- coords(pROC_graph, "best")
optimal_cutoff <- as.numeric(coords_output[1, "threshold"])

df_lasso_test <- df_lasso_test %>%
  mutate(Class_predicted = ifelse(pre_prob > optimal_cutoff, 1, 0))
# 1 = Rock, 0 = Electronic

# Test errpr
## Cutoff = 0.47
Class_test_lasso <- dplyr::mutate(
  df_lasso_test,
  Class_predicted = ifelse(
```

```
    pre_prob > optimal_cutoff, 1, 0  # 1 = Rock, 0 = Electronic
  )
)

tab <- table(Class_test_lasso$Y, Class_test_lasso$Class_predicted)
tab

confusionMatrix(
  factor(Class_test_lasso$Class_predicted),
  factor(Class_test_lasso$Y),
  mode = "everything",
  positive = "1"  # Assuming 1 = Rock is the positive class
)
```