STATS/CSE 780
Homework Assignment 2

Konrad Swierczek - 001423065

March 06, 2023

Introduction

Automated music genre classification tasks rely predominantly on audio feature extraction. The Echonest, purchased by Spotify in 2014, uses a set of 8 audio features for audio fingerprinting and genre classification, which is subsequently used for automated playlist development, and content-based reccomendation systems (Ellis et al., 2010). This study examinea the FMA (Free Music Archive, https://github.com/mdeff/fma) database, a free database for developing MIR (Music Information Retreival) (Defferrard et al., 2016) tools to evaluate The Echonest's audio features and their viability for music classification. Since The Echonest and subsequently Spotify have chosen to keep these features proprietary, investigating the relation between these features and subjective outcomes such as musical genre and popularity are one of the few ways to evaluate what these features signify. The FMA is a database containing 106,574 musical tracks and relevant audio features, metadata, and listening statistics. This study examines a subset of 13,129 tracks for which The Echonest's audio features are available.

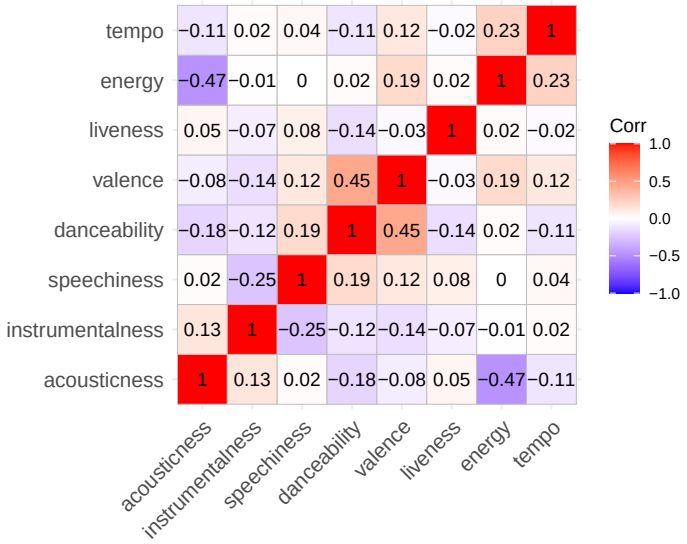


Figure 1: Correlation matrix for all The Echonest's audio features. Includes all 13,129 tracks from The Echonest subset of the FMA dataset.

Tracks missing top-level genre and popularity were excluded. Audio features were available for the entire subset of tracks. The dataset includes 12 categorical genres, 4 of which were excluded (see Supplementary Materials) from the genre analysis due to low sample size (<100). Further, a random sample of 241 tracks from each of the remaining genres was taken, as the smallest remaining genre (Jazz, 241 tracks) was sigificantly smaller than the largest genre (Rock, 3892 tracks). All

The Echonest's features are continuous variables with values between 0 and 1, with the exception of tempo which is a continuous BPM (Beats Per Minute) value ranging from 12.753 and 250.059. Figure 1 is a correlation matrix between each of The Echonest variables: as none of the correlation coefficients are higher than 0.5 or -0.5, the audio features do not appear to significantly covary. Figure 2 further outlines the distributions of each feature as they relate to each top-level genre.[1]

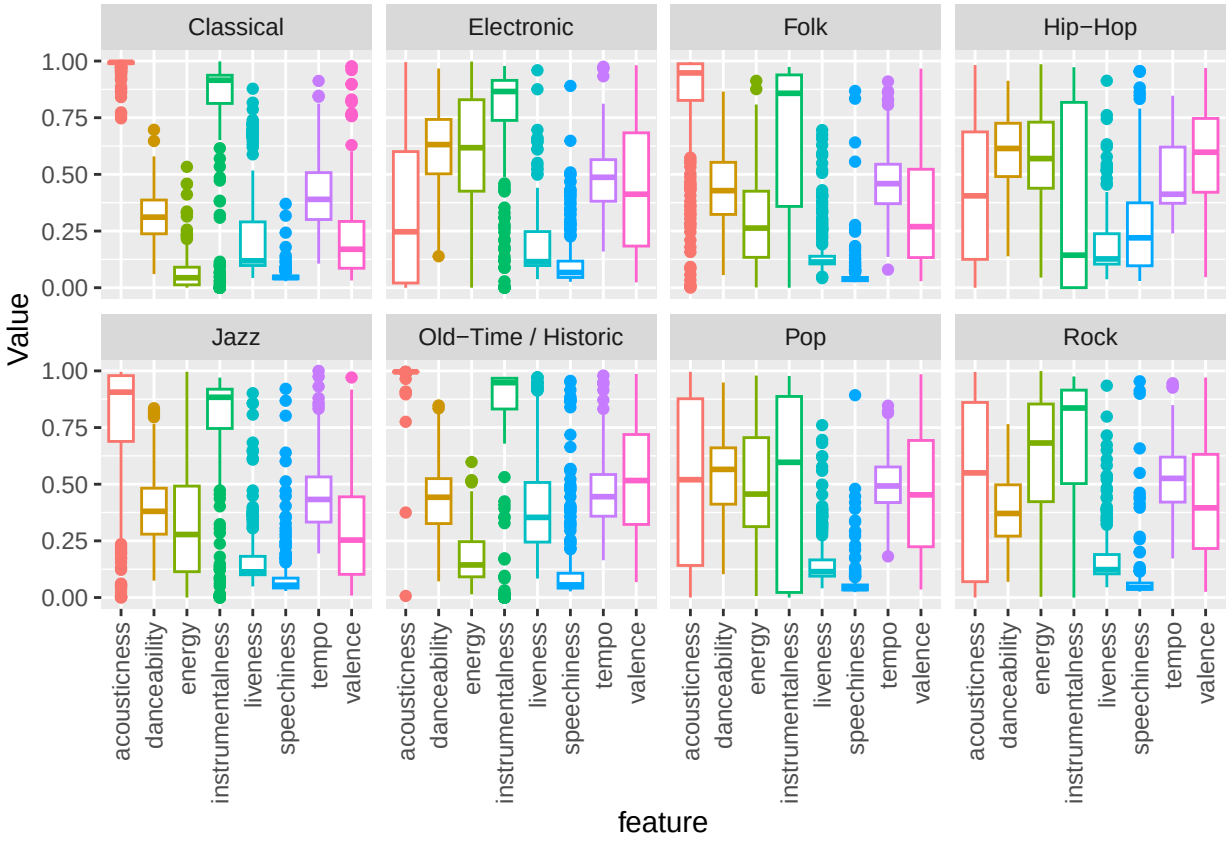


Figure 2: Boxplot distributions of The Echonest's audio features across genres. Tempo has been normalized to vary between 0 and 1.

Track popularity was derived from a play count variable indicating the amount of times a track was listened to: tracks with fewer listens than the median are assigned a n "unpopular" while tracks with more listens than the median are assigned a "popular" value. Outliers were not removed for any of The Echonest's audio features as variablity within a track might be responsible for differentiation in genre. Tracks with play counts outside of the 25% and 75% quantiles were removed (see Suplementary Materials for graphical summary). Multiple tracks appear to be significantly more played than the majority, likely due to social and economic factors rather than any pattern

[1] All materials and reproducible code used are available at https://github.com/konradswierczek/STATS780

in their audio features.

Methods

All analyses were performed in R (R Core Team, 2020).

Genre

Genre classification was performed using K-Nearest Neighbor. The dataset was split into two equal parts for training and testing, with equal samples of each genre in each set. Neighborhod size was determined using k-fold repeated cross validation, where the minimum error was evaluated as k = 9. (see Supplementary Materials).

Popularity

Logistic Regression and a Decision Tree were used to create models to predict whether a track is popular. Stratification (50-50) was performed seperatley for this analysis to ensure an equal split of popular and unpopular tracks occured. Finnally, logistic regression with shrinkage (lasso) was performed.

Results

Genre

KNN model training returned an accuracy of 0.4895833 when predicting the top genre of the training set. Given the subjective and often leaky nature of genre (Bansal et al., 2021), this performance indicates there may indeed be discrimination of genre or style from these audio features. However, further work is needed to understand how subjective genre preferences relate to audio features.

Popularity

Multiple logistic regression revaled a significant effect of all audio features except Valence and Tempo. Since Valence is commonly thought to relate to mood and tempo is often variable across

styles and genres, it is not surprising these variables are not predictive of popularity. Null deviance was greater than residual deviance, indicating these variables are good predictors of popularity. The most important predictors appear to be Speechiness (the presence of a voice), Liveness, and Danceability. However, prediction of the hold-out set was relatively poor. The miss-classification error rate was 0.4305279, and the sensitivity and specificty were respectivley 0.570609 and 0.5683326, indicating a prediction accuracy at approximatly chance. A Decision Tree analysis was also performed to predict popularity. The results of this analysis only used Acousticness with three terminal nodes to predict popularity, with a misclassifcation error of 40%. Overall accuracy to the test set was only 57%, indicating a poor fit.

Conclusions

Based on this analysis, it appears that The Echonest's audio features may be valuable features in genre classification as well as predicting track popularity. However, the goals of these tasks modulate the expectation of model performance. In the case of genre classification, accuracy aproaching 50% indicates that the genres examined here do differ in there audio features, leaving opportunities to improve these features and develop more precise prediction methods. Indeed, this sample is a small cross-section of genre and a genre classification model would likely perform better with more data. Further, genre may not be an ideal classification task due to its subjective nature. Popularity prediction from acoustic features appears to be less promising. Given the at-chance level of performance seen here, it may be the case that metadata features are predictive than audio features: this hypothesis is consistent with the dominance of metadata-based reccomendation systems. The measure of popularity used here also assumes the median is a suitable breaking point: other measures might be more suitable. Despite the limitations of this study, exploring audio features for MIR tasks is still a useful avenue of research to explore.

References

Bansal, J., Flannery, M. B., & Woolhouse, M. H. (2021). Influence of personality on music-genre exclusivity. *Psychology of Music*, *49*(5), 1356–1371.

Defferrard, M., Benzi, K., Vandergheynst, P., & Bresson, X. (2016). FMA: A dataset for music analysis. *arXiv Preprint arXiv:1612.01840*.

Ellis, D. P., Whitman, B., Jehan, T., & Lamere, P. (2010). *The echo nest musical fingerprint*.

R Core Team. (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Supplementary Materials

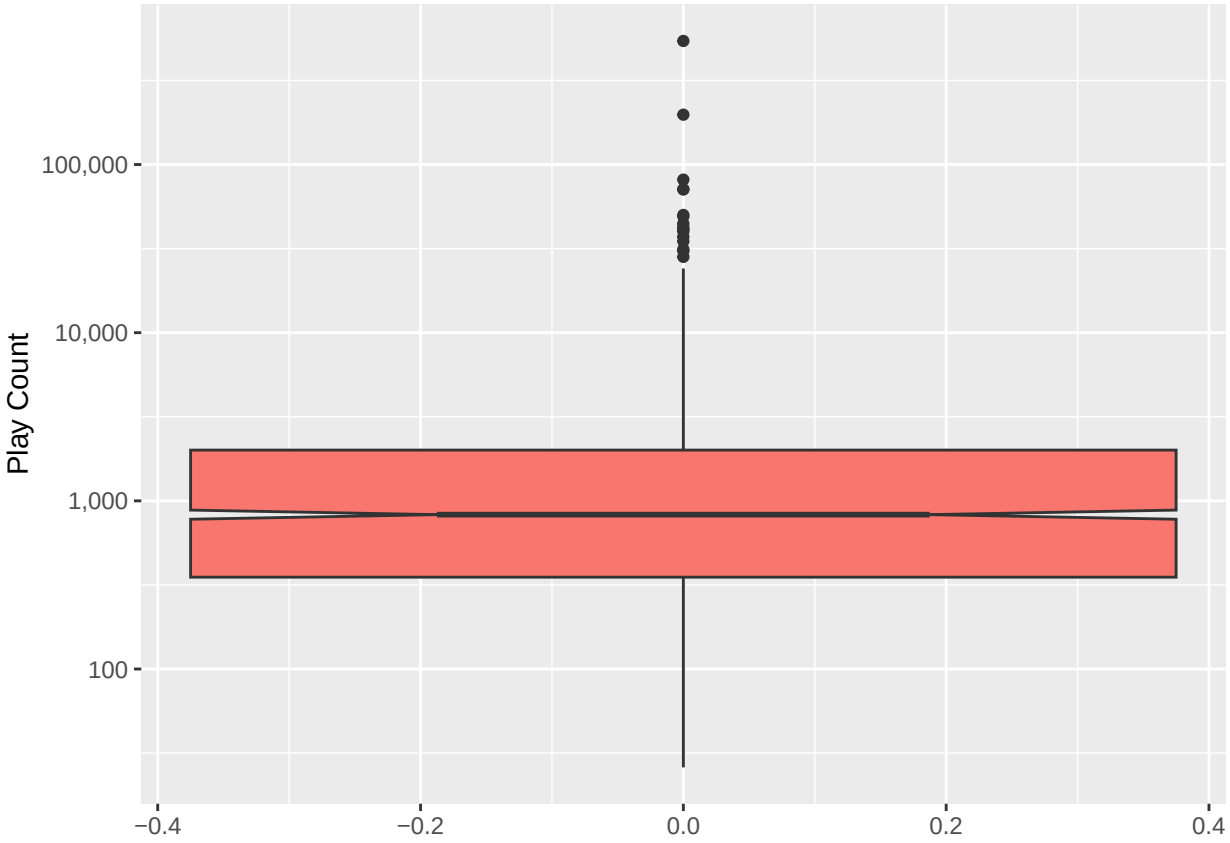


Figure 3: Boxplot distribution of play counts, used to derive a popularity measure. Values below the notch were labelled 'unpopular', while values above were labelled 'popular'. Outliers were filtered out due to their dissproportionate impact on the analysis.

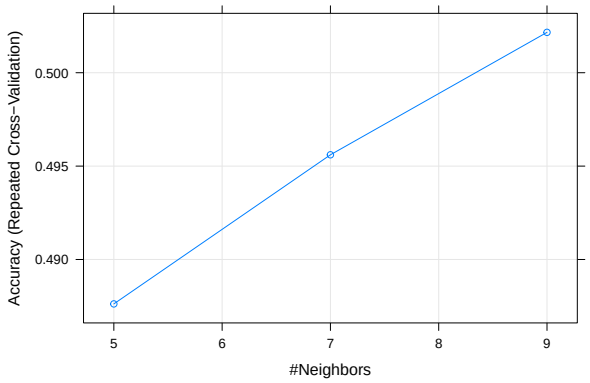
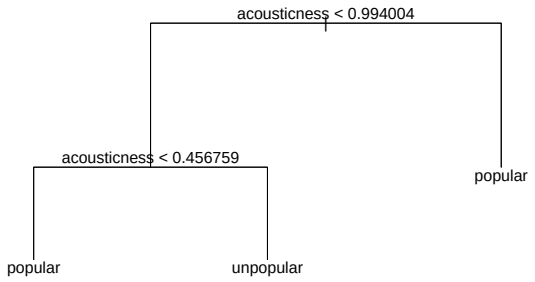


Figure 4: Results of k-fold cross validation for knn algorithm. the chosen value of K was 9.



\begin{figure}[H] \caption{Decision Tree analysis results: only one audio feature, acousticness, was included by the algorithm. Accuracy to the test set was 58%} \end{figure}

```
##
## Call:
## glm(formula = popularity ~ acousticness + danceability + energy +
##     instrumentalness + liveness + speechiness + tempo + valence,
##     family = binomial("logit"), data = pop_train[, -1:-2])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7814  -1.1418   0.6741   1.1425   1.8012
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.1113400  0.1875561  -0.594  0.55276
## acousticness      0.7462065  0.0938492   7.951 1.85e-15 ***
## danceability     -0.9631392  0.1896694  -5.078 3.81e-07 ***
## energy            1.2076545  0.1295189   9.324  < 2e-16 ***
## instrumentalness -0.2680584  0.0903874  -2.966  0.00302 **
## liveness         -1.2169471  0.1939987  -6.273 3.54e-10 ***
## speechiness       1.0422268  0.2352023   4.431 9.37e-06 ***
## tempo            -0.0007418  0.0008944  -0.829  0.40693
## valence          -0.2061209  0.1305058  -1.579  0.11424
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6277.1  on 4527  degrees of freedom
## Residual deviance: 6073.5  on 4519  degrees of freedom
## AIC: 6091.5
##
## Number of Fisher Scoring iterations: 4
```

```
# knitr setup
knitr::opts_chunk$set(echo = FALSE, warning = FALSE, message = FALSE)
knitr::opts_chunk$set(fig.pos = "H", out.extra = "")
# Imports
packages <- c("ggplot2", "readr", "dplyr", "knitr", "tidyr", "gridExtra",
              "ggcorrplot", "class", "glmnet", "plotROC", "pROC", "tree", "caret")
lapply(packages, library, character.only = TRUE)
# Set Seed
set.seed(75)
# 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("\\...*", "", .)
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)
```

```r
# Tidy Data
tidy_data <- data[, c(41, 48, 54, 55, 56, 57, 58, 59, 60, 61)] %>%
# Dropping rows with missing genre
  drop_na(., track_genre_top) %>%
# Dropping four genres with lowest sample size
  dplyr::filter(!track_genre_top %in% c("Experimental", "Blues",
                                        "Instrumental", "International"))
# Creating a binary variable for popularity based on data median
tidy_data$popularity <- ifelse(tidy_data$track_listens >
                                 median(tidy_data$track_listens),
                                 "popular", "unpopular") %>%
                         as.factor()
# Write .RData
save(tidy_data, file = "assignment2/tidy_data.RData")
# Load data parsed above to minimize linting time.
load("tidy_data.RData")
# Should include some exploratory stuff?
# Sample 241 observations from each genre
sampled_data <- tidy_data %>%
                group_by(track_genre_top) %>%
                sample_n(241) %>%
                ungroup()
# Data split for genre
knn_index <- caret::createDataPartition(sampled_data$track_genre_top,
                                        p = 0.5, list = FALSE)
knn_train <- slice(sampled_data, knn_index)
knn_test <- slice(sampled_data, -knn_index)
# Data split for popularity
pop_index <- caret::createDataPartition(tidy_data$popularity,
                                        p = 0.5, list = FALSE)
pop_train <- slice(tidy_data, pop_index)
pop_test <- slice(tidy_data, -pop_index)
```

```r
# Correlation Matrix
cor_mat <- round(cor(tidy_data %>% dplyr::select(-1, -2, -11)), 3)
ggcorrplot(cor_mat, hc.order = TRUE, lab = TRUE)
# Function to normalize Tempo
cent_norm <- function(x, na.rm = T) (x  / max(x, na.rm = T))
ggplot(sampled_data %>%
        mutate(tempo = cent_norm(tempo)) %>%
        gather(feature, val, 3:10),
        aes(x = feature, y = val, colour = feature)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  theme(legend.position = "none") +
  ylab("Value") +
  facet_wrap(~ track_genre_top, nrow = 2)
# KNN with K-fold repeated cross-validation for Genre Classification.
trctrl <- trainControl(method = "repeatedcv", number = 15, repeats = 3)
knn_fit <- train(track_genre_top ~ acousticness + danceability + energy +
                                   instrumentalness + liveness + speechiness +
                                   tempo + valence,
                 data = knn_train, method = "knn", trControl=trctrl,
                 preProcess = c("center", "scale"))
knn_pred <- predict(knn_fit, newdata = knn_test)
knn_y_test <- dplyr::pull(knn_test, track_genre_top)
knn_mean <- mean(knn_pred == knn_y_test)
# Multiple Logistic Regression for Popularity
pop_logreg <- glm(popularity ~ acousticness + danceability + energy +
                               instrumentalness + liveness + speechiness +
                               tempo + valence,
              data = pop_train[, -1:-2], family = binomial("logit"))
pred_prob <- predict(pop_logreg, pop_test, type = "response")


pop_logreg_stat <- tibble(
```

```
  pre_prob = pred_prob,
  Y = pull(pop_test, popularity))
cutoff <- 0.5
pop_logreg_stat <- dplyr::mutate(
  pop_logreg_stat,
  Class_predicted = ifelse(pre_prob > cutoff, "unpopular", "popular"))
tab <- table(pop_logreg_stat$Y, pop_logreg_stat$Class_predicted)
# miss-classification error rate
miss <- 1- sum(diag(tab))/sum(tab)
# sensitivity
sens <- tab[2,2]/sum(tab[2,])
# specificity
spec <- tab[1,1]/sum(tab[1,])
# Decision Tree on popularity
tree_data <- tree(popularity ~ acousticness + danceability + energy +
                               instrumentalness + liveness + speechiness +
                               tempo + valence,
                               data = pop_train)
tree.pred <- predict(tree_data, pop_test, type = "class")
tree_acc <- (table(tree.pred, pop_test$popularity)[1,1] +
table(tree.pred, pop_test$popularity)[2,2]) /
sum(table(tree.pred, pop_test$popularity))
X <- model.matrix(popularity ~ acousticness + danceability + energy +
                               instrumentalness + liveness + speechiness +
                               tempo + valence,
                               data = pop_train)[, -1]
Y <-  dplyr::pull(pop_train, popularity)
X_test <- model.matrix(popularity ~ acousticness + danceability + energy +
                               instrumentalness + liveness + speechiness +
                               tempo + valence,
                               data = pop_test)[, -1]
```

```
cv_lasso <- cv.glmnet(
  X,
  Y,
  family='binomial',
  alpha = 1,
  standardize = FALSE
  )

test_set_pred_log_odds <- predict(
  cv_lasso,
  newx = X_test,
  s = "lambda.1se"
  )

test_set_pred_prob <- exp(test_set_pred_log_odds)/(1+exp(test_set_pred_log_odds))

df_lasso_test <- tibble(
  pre_prob = test_set_pred_prob,
  y = pull(pop_test, popularity)
  )

#ggplot(
#  df_lasso_test,
#  aes(m = pre_prob, d = y)) +
#  geom_roc(n.cuts=20, labels=FALSE)
#rocgraph +
#  style_roc(theme = theme_grey) +
#  geom_rocci(fill="pink")
ggplot(sampled_data, aes(y = track_listens, fill = "red")) +
  geom_boxplot(notch = TRUE) +
  scale_y_continuous(trans = "log10", labels = scales::comma) +
```

```
  theme(legend.position = "none") +
  ylab("Play Count")
plot(knn_fit)
plot(tree_data)
text(tree_data, pretty = 0)
summary(pop_logreg)
```