Predicting song popularity by genre

Kathy Li, Sa Qu, Rulan Xiao

Project overview and motivation

- Our aim was to utilize **audio features** and **track metadata** to predict **song popularity**
- Our data is from the Free Music Archive, https://freemusicarchive.org/

Input

Audio features: acousticness, danceability, energy, etc.

Track metadata: track title, duration, bit rate, location

Classification model

(Logistic regression, KNN, decision tree, random forest)

Output

Popular or not popular

Data overview

- **Audio features:** speechiness, liveness, energy, tempo, etc.
- **Track features:** title, latitude / longitude, duration, bit rate, etc.
- Additional feature: sentiment analysis on song name - we take positive and negative sentiment using nltk package
- We find the year with the most tracks (2010) and select the top 3 genres (rock, hip-hop, electronic)
- To change our data to classification, we take the **top 25% of track listens to be popular** and the rest as unpopular

Sentiment analysis

Code

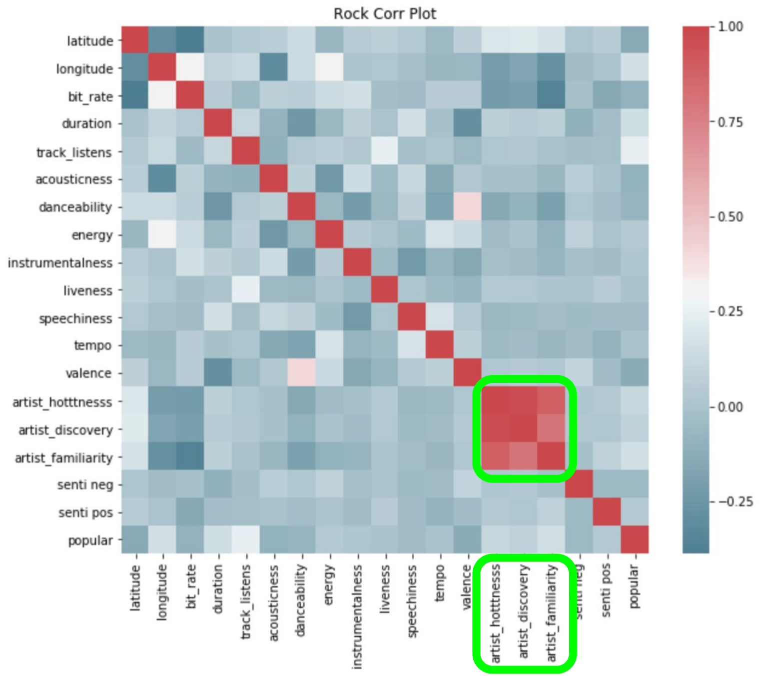
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')

# Read in track data - change to local directory
tracks_senti = pd.read_csv('/Users/kathyli/Downloads/fma_metadata/tracks.csv',header=None)

header = tracks_senti.iloc[1]
header[0]='track ID'
header[52]='track title'
tracks_senti.drop(tracks_senti.index[[0,1,2]],inplace=True)
tracks_senti.rename(columns = header,inplace=True)
tracks_senti.head()

df=tracks_senti[['track ID','track title']]
df.dropna(axis=0, how='any')

ml = df["track title"].values
title=[]
for i in range(len(ml)):
    a=str(ml[i])
    title.append(a)
idd = df["track ID"].values

neg=[]
neu=[]
pos=[]
comp=[]

sid = SentimentIntensityAnalyzer()
for sentence in title:
    ss = sid.polarity_scores(sentence)
    score=[]
    for k in ss:
        a=ss[k]
        score.append(a)
    neg.append(score[0])
    neu.append(score[1])
    pos.append(score[2])
    comp.append(score[3])
```

Output

	senti comp	senti neg	senti neu	senti pos	track_ID	track_title
0	0	0	1	0	2	Food
1	0	0	1	0	3	Electric Ave
2	0	0	1	0	5	This World
3	0.0516	0	0	1	10	Freeway
4	0	0	1	0	20	Spiritual Level
5	0.6369	0	0.417	0.583	26	Where is your Love?
6	0.5719	0	0.213	0.787	30	Too Happy
7	0	0	1	0	46	Yosemite
8	0	0	1	0	48	Light of Light
9	0	0	1	0	134	Street Music
10	0	0	1	0	135	Father's Day
11	0	0	1	0	136	Peel Back The Mountain Sky
12	0	0	1	0	137	Side A
13	0	0	1	0	138	Side B
14	0	0	1	0	139	CandyAss
15	0	0	1	0	140	Queen Of The Wires
16	0	0	1	0	141	Ohio
17	-0.3818	0.565	0.435	0	142	Punjabi Watery Grave
18	0	0	1	0	144	Wire Up
19	0	0	1	0	145	Amoebiasis
20	0	0	1	0	146	Gimme a Buck or I'll Touch You / Boilermaker
21	-0.25	0.5	0.5	0	147	Repetitive Motion Sickness
22	0	0	1	0	148	Blackout 2
23	0	0	1	0	149	Outside the window bees buzzed...
24	0.4939	0	0.686	0.314	150	... listening to the sunshine burn the grass
25	0	0	1	0	151	Untitled 04
26	0	0	1	0	152	Untitled 11
27	0	0	1	0	153	Hundred-Year Flood
28	0	0	1	0	154	Squares And Circles
29	0	0	1	0	155	Maps of the Stars Homes
30	0	0	1	0	156	Track 01
31	0	0	1	0	157	Track 02
32	0	0	1	0	158	Track 03
33	0	0	1	0	159	Track 04

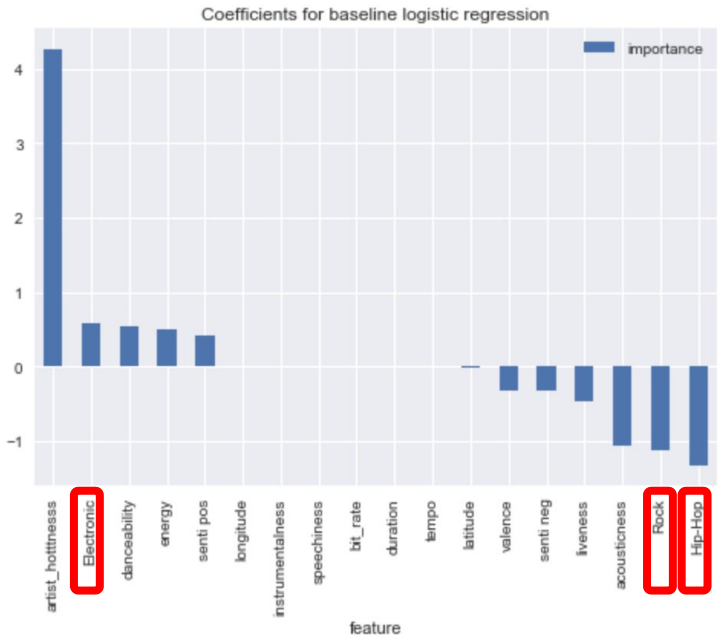
Exploratory data analysis

- We plot correlation between variables and see that artist hottness, artist familiarity, artist discovery are redundant, so we choose one
- We also standardize data using StandardScaler



Baseline model

- Our hypothesis: **genre is an important predictor for song popularity**
- To validate this, we first implement a baseline logistic regression, where we include **genre as a categorical variable** (rock, hip-hop, electronic)
- **Genre variables have high magnitude coefficients**, so we subset by genre to investigate what separates the genres



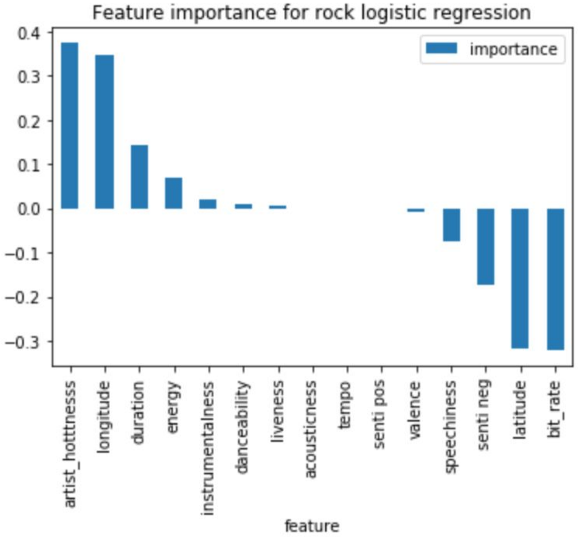
Genre-specific models

- For each genre, we test
  - **Logistic regression** - regularize using L1 penalty
  - **Decision tree** - find optimal tree depth
  - **Random forest** - find optimal depth and max number of features
  - **KNN** - find optimal number of neighbors
- We perform five-fold cross-validation and for each genre, pick the model that performs optimally on random subsets based on AUC ROC
- We also look at **feature importance** to determine which features are most important for each genre

Logistic regression for rock example

Logistic regression with L1 penalty

```
# Regularize over L1 penalty
C_vals = np.logspace(-4,0,100)
scores = []
for C_val in C_vals:

    #change penalty to l1
    regr = LogisticRegression(penalty='l1', C = C_val)
    regr.fit(X_train, y_train)

    probas_ = regr.fit(X_train, y_train).predict_proba(X_test)

    fpr, tpr, thresholds = roc_curve(y_test, probas_[:, 1])
    roc_auc = auc(fpr, tpr)

    scores.append(roc_auc)

#plot alphas vs. scores
plt.plot(C_vals, scores)
plt.xlabel('alpha')
plt.ylabel('AUC ROC')
plt.title('alpha vs. AUC ROC')
plt.show()

C_best_L1 = C_vals[scores.index(max(scores))]
print('Optimal C value for logistic regression: ' + str(C_best_L1))
```



Artist hotness, longitude/latitude, duration, and bit rate are important coefficients for rock

Web app demo

rlx.pythonanywhere.com

Input: song info

Output: popular / not popular