Song Mood Classification

Classifying songs into happy or sad using Spotify's audio features

By Ryan Moerer

Motivation

- Try my hand at song classification.
- Gain insight into what characteristics of a song contribute to making it happy or sad.
- Fit, tune, and compare several popular machine learning models.
- Be able to make accurate predictions about whether a song is happy or sad.

Data Collection

Spotify API

- Spotify's API provides a variety of audio features and information for each track in its collection:
    - Acousticness
    - Danceability
    - Instrumentalness
    - Energy
    - Speechiness
    - Valence (Musical positivity)
    - Liveness (Likelihood that a track is live)
    - Loudness
    - Key (Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on)
    - Mode (1 = Major, 0 = Minor)
    - Tempo
    - Time Signature (ranges from 3 to 7 indicating time signatures of "3/4", to "7/4")
    - Explicit
    - Duration

Challenges

- Finding a sample of songs accurately labelled happy and sad.
- Creating a representative sample of happy and sad songs.
- Combining various Spotify API calls into a single dataframe.

Data Collection Method

- Get a subset of 50 playlists entitled “happy songs” and “sad songs” respectively.
- Get all track id’s for all playlists.
- Get track meta-information (name, artist, album, etc.).
- Get audio features for each track.
- Combine all information into single dataframe.
- Create balanced dataset.

Dataset

- 4555 Happy songs
- 4555 Sad Songs
- y = {1: happy, 0: sad}

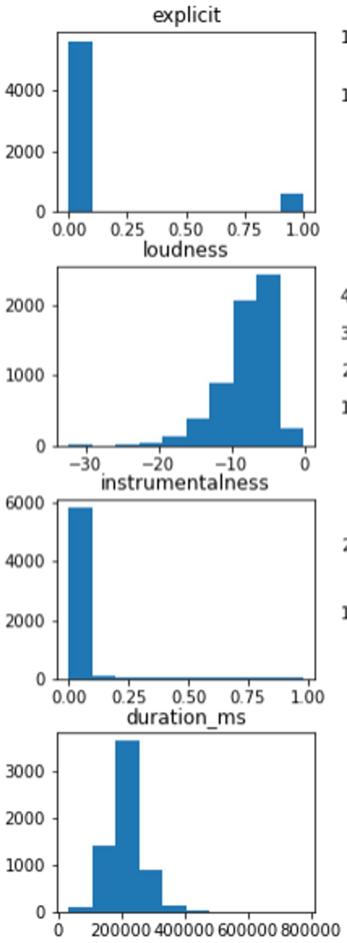
playlist_name	Sad Songs
playlist_id	37i9dQZF1DX7qK8ma5wgG1
name	Bedroom Ceiling
album_name	Bedroom Ceiling
artist	Sody
id	1jXKRARVJQFG3jTvZEgxsP
popularity	55
explicit	False
danceability	0.49
energy	0.416
key	6
loudness	-7.702
mode	1
speechiness	0.127
acousticness	0.728
instrumentalness	0
liveness	0.0694
valence	0.236
tempo	125.928
duration_ms	180086
time_signature	4
y	0

Data Cleaning & Pre-Processing

- Only include information relevant to the musical characteristics of the track.
- Remove unnecessary/noisy features.
- One-hot encoding categorical data?
    - Mode and explicit already encoded as binary features.
    - Key already ordinally ranked.
- Split dataset into testing and training set (70% training, 30% testing, stratify=True).
- Further preprocessing/variable transformation done on model by model basis.

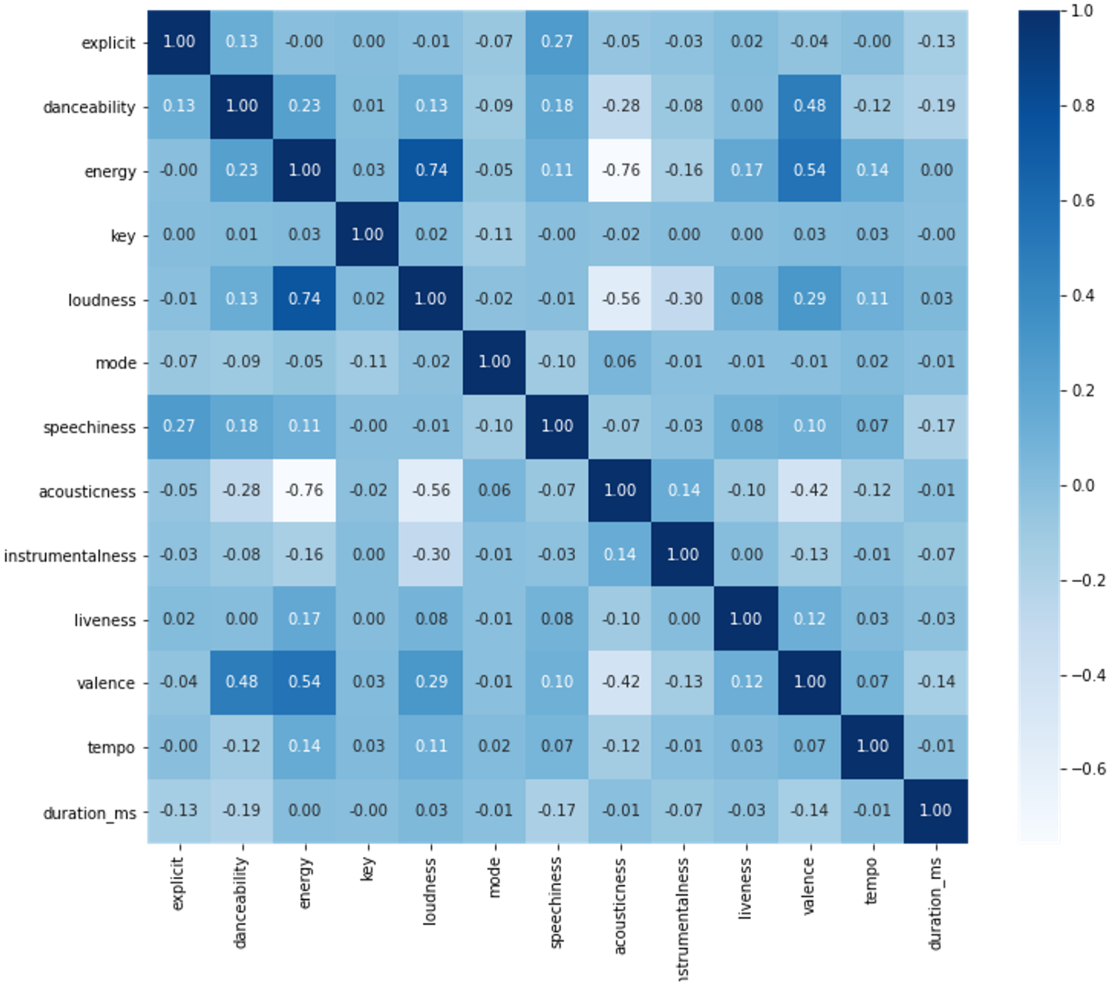
explicit	False
danceability	0.545
energy	0.78
key	7
loudness	-4.867
mode	0
speechiness	0.0436
acousticness	0.0309
instrumentalness	4.64e-05
liveness	0.0828
valence	0.458
tempo	125.014
duration_ms	255093

Data Exploration

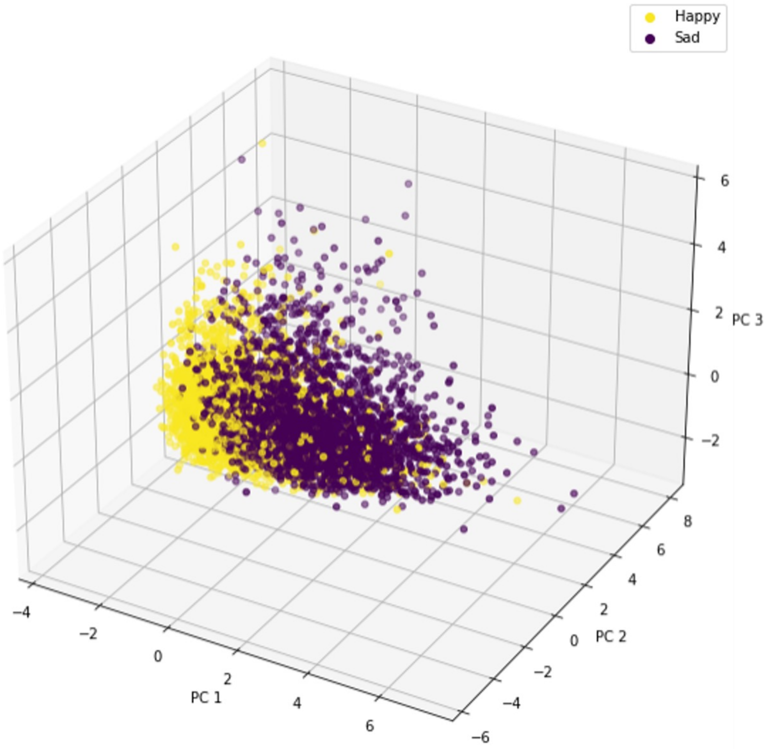


explicit
danceability
energy
key
loudness
mode
speechiness
acousticness
instrumentalness
liveness
valence
tempo
duration_ms

Correlation Matrix of Features



3D PCA Plot of Features



Modeling

Methodology

- Model choices:
    - K Nearest Neighbors
    - Logistic Regression
    - Random Forest
        - An ensemble approach to modelling that fits a multitude of decision trees and outputs the class selected by the most trees.
- Use cross-validation and grid search to tune hyperparameters.
- Metric of interest: Accuracy

K Nearest Neighbors

```
# standardize the training features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
# create param grid
params = {'n_neighbors': list(range(1,51))}
# KNN classifier with weights computed as inverse of distance
knn = KNeighborsClassifier(weights='distance')
# grid search to tune k
grid_search_knn = GridSearchCV(knn, param_grid=params, scoring='accuracy', cv=10)
grid_search_knn.fit(X_train_scaled, y_train)
# best training accuracy and params
knn_gridsearch.best_params_, knn_gridsearch.best_score_
({'n_neighbors': 21}, 0.832453080627238)
```

Logistic Regression

```
# create param grid
params = {'C':10**np.linspace(-4,5,20)}

# Logistic Regression with l1 regularization
log_reg = LogisticRegression(penalty='l1', solver='liblinear', random_state=42)

# grid search to tune penalty parameter
log_gridsearch = GridSearchCV(log_reg, params, scoring='accuracy', cv=10)
log_gridsearch.fit(X_train_scaled, y_train)

# best params/best cross-validated accuracy
log_gridsearch.best_params_, log_gridsearch.best_score_
({'C': 0.023357214690901212}, 0.8350192410585671)
```

Logistic Regression Coefficients

valence	1.233662
energy	0.843703
mode	0.160252
danceability	0.138826
liveness	0.019129
tempo	0.000000
instrumentalness	0.000000
key	0.000000
speechiness	-0.075846
duration_ms	-0.144780
loudness	-0.326367
explicit	-0.329135
acousticness	-0.556455

Random Forest

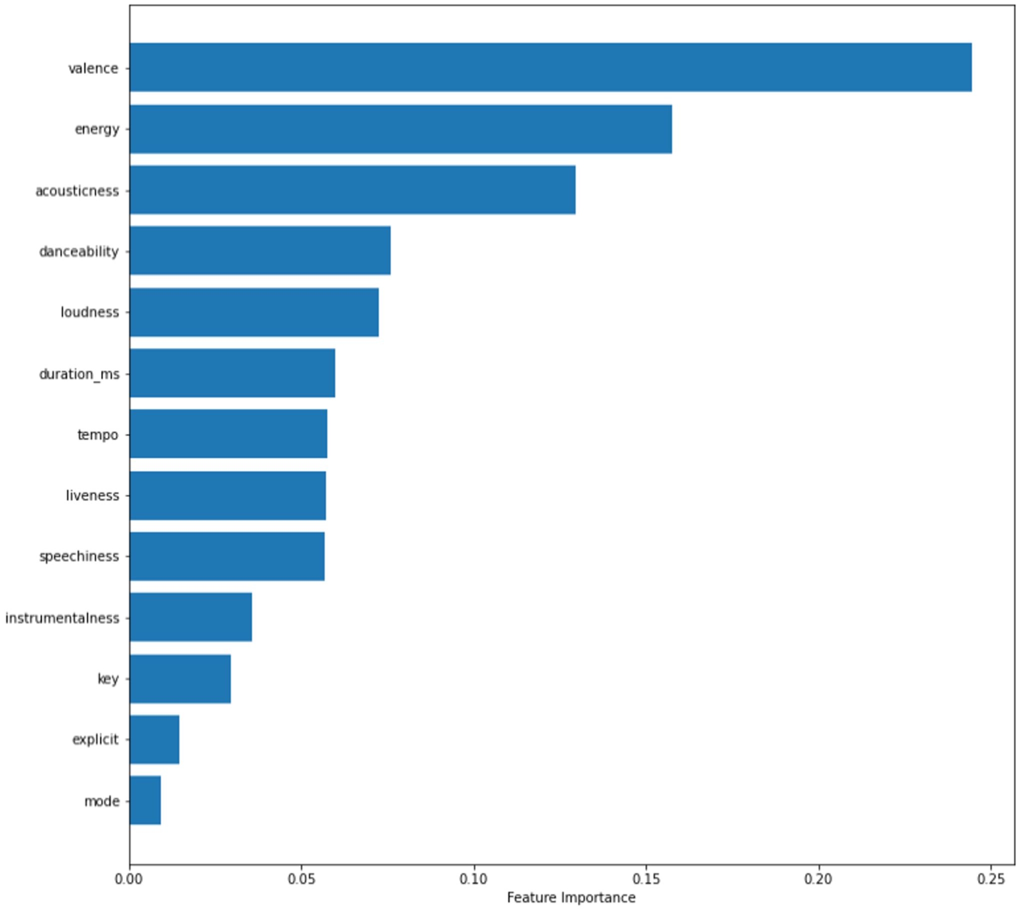
```
# param grid
params = {'max_features': [1,2,3,4,5,6]}

# RF model with 1000 trees
rf_clf = RandomForestClassifier(n_estimators=1000,random_state=42)

# gridsearch to find best number of features to sample at each split
rf_gridsearch = GridSearchCV(rf_clf, param_grid=params, cv=5)
rf_gridsearch.fit(X_train, y_train)

# best params/cross-validate accuracy
rf_gridsearch.best_params_, rf_gridsearch.best_score_
({'max_features': 2}, 0.8435143061295829)
```

Feature Importance



Test Set Results

Model	Precision	Recall	F1 Score	Accuracy
Random Forest	0.836	0.826	0.831	0.832
Logistic Regression	0.834	0.803	0.817	0.821
KNN	0.820	0.803	0.812	0.814

Happy

	name	artist	happy_prob	y
0	Accidentally In Love - From "Shrek 2" Soundtrack	Counting Crows	0.994	1
1	Valerie (feat. Amy Winehouse) - Version Revisited	Mark Ronson	0.996	1
2	Tamma Tamma Again	Bappi Lahiri	0.994	1
3	Classic	MKTO	0.999	1
4	Good Time	Owl City	0.993	1
5	Move Your Feet	Junior Senior	0.995	1
6	Bad Vibrations	Val Astaire	0.995	1
7	You Drive Me Crazy - (Single Version) [Remaste...	Shakin' Stevens	0.993	1
8	Out of My League	Fitz and The Tantrums	0.994	1
9	Don't Slack (from Trolls World Tour)	Anderson .Paak	0.993	1

Sad

	name	artist	sad_prob	y
0	Rainbow	Josh Rabenold	0.9922	0
1	Skinny Love	Birdy	0.9960	0
2	High Hopes	Kodaline	0.9980	0
3	Behind the page	Kwon Jin Ah	0.9970	0
4	Skinny Love	Birdy	0.9960	0
5	Bad Day	IU	0.9940	0
6	Wish	Urban Zakapa	0.9900	0
7	Love Like This - Acoustic	Kodaline	0.9940	0
8	DOWN	PARK WON	0.9930	0
9	lovely	Billie Eilish	0.9930	0

Areas For Improvement

- Include other moods.
- Larger, more representative sample of data.
- Include other characteristics of songs (Song title, lyrics, artist, etc.).
- Spend more time on tuning hyperparameters.
- Try smarter/different models