BigData&DataAnalytics

Spotify Prediction base on Machine Learning Algorithms

Mauricio Carvajal
IOT ANALYTICS

Table of contents

1. Introduction

The following project consit in aply all the learnings in data science in a real personal project. In summary this process has been excellent due to there is no plan of attack, which forces to read, investigate and apply the best practices.

It's important to mention that for this project we use 3 models that are familiar with the course, and there is try to start using neural networks algortim, in which this can be dicussed in the following document, as part of the investigation.

2. Justification of the project

The following project consist in create a Spotify Predcition is a song will be like or not base on machine learning algoritms. The spotify plataform has been choosen due to this popularity wordwide, as also the positiliby of use API to extract the feactures.This is very important due provide standard method to classify the songs.

Spotify besides been very popular, also have a big data base. So the pilot project consist in the predicitons using supervised algorithms, but will be open for future steps to develop unsupervised.

3. Libraries

The following project has been develop using python and very importand libraries,in the Jupyter Notebook and in the anexos of the following documents are listed all of them, but as summary we used:

Data manipulation

- **Pandas**: to handle the data sets
- **Numpy**: Provides a high-performance multidimensional array object, and tools for working with these arrays.

Visualization

- **Matplotlib**
  - It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTL+
- **Seaborn**
  - Seaborn is a Python data visualization library based on matplotlib.

Machine Learning on Python

- **Sklearn**: is a free software machine learning library for the Python programming languag
  - **Preprocessing**
    - StandardScaler: Standardize features by removing the mean and scaling
    - LabelEncoder: Used to normalize labels
  - **Classification**
    - RandomForestClassifier
    - KNeighborsClassifier
    - SVC: Support vector machine
  - **Cross Validation**

    - Train_test_split: Split arrays or matrices into random train and test subsets
- **Metrics**:
    - mean_squared_error: Mean squared error regression loss
    - r2_score: R^2 (coefficient of determination) regression score function.
    - cross_val_score: Evaluate a score by cross-validation
    - classification_report: Build a text report showing the main classification metrics
    - confusion_matrix: Compute confusion matrix to evaluate the accuracy of a classification.
    - accuracy_score: Accuracy classification score.
    - cohen_kappa_score: Cohen's kappa: a statistic that measures inter-annotator agreement.
- **PCA**
    - Principal component analysis (PCA).
        - Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space.
- **OneHot econder**
    - Encode categorical features as a one-hot numeric array.

4. Data Science Framework used

The Process that has been follow in this project is shown below, it's important to mention that all the best practices have been incorporate in order to maximize the effectiveness of the process. The framework used is go from the understding the data, the preprocess and exploratory analsys, all the steps that has been takin to develop the model, improve it and evaluated in order to give a prediction.

The steps are shown below with a brief explation, for further details please go to the Jupyter Notebooks.

4.1 Load and Cleaning

This is one of the first step, in which is needed to upload all the data and concatened in 1 data frame that will be needed to continue the pre process. IIn this steps start with data cleaning, deleting information that is not needed, searching from missing values.

4.2 Preprocesing the data

- Feature Understanding
    - Understand the data, what kind data, how big, missing, representative data is fundamental.
- Tranforming data
    - After the feature understanding, is needed to transform the data to be properly used. i.e numerical to categorical.
- Binning data
    - Binning the data is the porcess in which the data is transform to reduce the effects of minor observation errors
- Scalate data

        - Binning the data is the porcess in which the data is transform to reduce the effects of minor observation errors

4.3 EDA (Exploratory Data Analysis)

EDA, or Exploratory data analsys is a critical step in data science, due to this describe a high level how is the data, what are the relevant features, if the data is enough, and a lot of eleement that provide a overview.

This step is needed previos to build a machine learning model, some of the activities that involves EDA are:

- Visualization and Statistics about each variable
- Scatter plots comparing the relationships between any two variables

4.4 Feature Engineering

In feature engineering, it can be defined as the process to improve the features that will be used for the machine learning algorithms or data mining algorithms. This process will take the database, analyzed, split it, pre-process, transformed and then presented to be ready to used to build the algorithm.

- Feature Selection
    - In this spet we split the data frame work in order to preprocess before build the the predictions models.
- One Hot Encoder
    - One Hot Encoder help to improve the performance of the buld algo due to help to move a categorical value to be represented into numerical.
    - The main different between One hot Encoder and lable encoder, is that the traditional label encoder convert the categorical data into numerical using 1, 2, 3. This can be issue for the model, due to can understood this 3 value as more importnat or more relevant that 1.
- PCA
    - Principal component analysis (PCA). Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space. But after the tunning and the iteration, the PCA was not needed for this dataset.

4.5 Model Development

In this section we use different classifications models, we evaluate and the tunning of each to them be able to evaluate each on them and chose the one with the best performance.

For the present project, the models that are been used are base on the type of problem that is needed to address. In this case, is needed to predict if a certain song will be like by the users. This is clearly a classification problem, base on this the algorithms that are been elected are:

4.5.1 Random Forest

The first algorithm that is been chosen is Random Forest, due to it has a good performance in datasets of middle size and in a classification problem,

Random forests can be defined as an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes

4.5.2 Support vector Machine

Support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text.

4.5.3 KNN

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

4.5.4 Neural NetworkModel Prediction

Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function by training on a dataset, where is the number of dimensions for input and is the number of dimensions for output. Given a set of features and a target , it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers.

4.6 Model Evaluation

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called overfitting. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a test set X_test, y_test.

4.6.1 cross_val_score

Cross_val_score is the simplest way to use cross-validation.

4.6.2 accuracy_score

Acuracy classification score.In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

4.6.3 cohen_kappa_score

Cohen's kappa: a statistic that measures inter-annotator agreement.

4.6.4 confusion_matrix

Compute confusion matrix to evaluate the accuracy of a classification.

4.6.5 classification_report

Build a text report showing the main classification metrics

4.5.6 Model Score

We use the .score to evaluate waht is the best model

5. Results

In this section we present the results that has been obtain after following the framework explained adove. The results consist in show what are the insights on the data that has been used, learnings from the preprocessing the data, the modetls that has been build and the results that has been obtained.

It's important to mention that in this section we justify what are the model that has been choosen, also all the recommendations for improvmements.

5.1 Model build Insights

As in the previous sction the has listed all the steps that has been follow to build and evaluate the model, but here are the insighta of follow up the stepts:

- Cleaning the data is critical, in order to ajust the correctness of the model, for example, chaning numerical values into categorical, process the missing values, removing features that are not relevant.
- OnehotDecoder is very usefull when there is categorical data that needs to convert in numerical, but is desire to eliminate any possible bias that can cause having big numbers angains the minor ones.
- In the case of preprocessing the data, this helps to improve the performance of the modelts, in this project we used to scalated the data, grouping or binning in order to understand better the behabiour.
- EDA (Exploratory data analysis), this taks is also critical to understand the data, and the differentet feature and start to having idea the level of importance. From this section there a lot of visualizations that has been used to understand how the different features are relate.
- Feature enginnering is vital, in this project we used PCA that allows to summarize and to visualize the information in a data set containing individuals/observations described by multiple inter-correlated quantitative variables.
- In this project 3 typical machine learnigns algorints has been used (SVM,KNN, RF) and also included a neural network classifier MLP Multi-layer Perceptron. The there isnot big different using neural network agains machine learning, due to the data set has been used is relative small, also the number of feature are also relative small, but was important to have another reference point,

5.2 Models Restuls

In this Report all the technical information is placed in the "Jupyter Notebook", but something important is to present the results of the models that we used.

As shown in the picture all present similar behavior, but "RF" present better score, so for this project this is the one that is choosen. But the relevant information is the neural network model that not shown better results, and some insights about is:

- We need to do more turn in
- Those models perfrom better with large data sets

- Predict a human behabior is very difficult
- In music there is tendency related to feeling on the moment of the person

8.1 cross_val_score

Evaluate a score by cross-validation https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

Cross_val_score is the simplest way to use cross-validation.

```
In [355]: #Random Forest model
print(cross_val_score(modelRF, X_train, y_train))
[0.7940552  0.75583864 0.74200426]

In [356]: #Support Vector Machine
print(cross_val_score(modelSVM, X_train, y_train))
[0.75583864 0.73460722 0.71641791]

In [357]: #KNN
print(cross_val_score(modelKNN, X_train, y_train))
[0.70912951 0.67940552 0.67803838]

In [358]: # Neural network
print(cross_val_score(modelMLPC, X_train, y_train))
[0.7537155  0.72611465 0.71641791]
```

8.2 Model Score

We use the .score to evaluate waht is the best model

```
In [359]: #Random Forest model
modelRF.score(X_train,y_train)
Out[359]: 0.9985825655563431

In [360]: #Support Vector Machine
modelSVM.score(X_train,y_train)
Out[360]: 0.8086463501063076

In [361]: #KNN
modelKNN.score(X_train,y_train)
Out[361]: 0.7647058823529411

In [362]: # Neural network
modelMLPC.score(X_train,y_train)
Out[362]: 0.8015591778880227
```

5.3 Model Choosen

In this section, it's discussed the model that has been chosed for this project. In this case the model that present the best performance is the RF model.

8.4.1 Model Evaluation for RF

```
In [392]: print("Classification Report")
print(classification_report(y_test, predictionsRF))

Classification Report
              precision    recall  f1-score   support

           0       0.78      0.57      0.66       299
           1       0.67      0.85      0.75       307

    accuracy                           0.71       606
   macro avg       0.72      0.71      0.70       606
weighted avg       0.72      0.71      0.70       606
```

Classification Report held to build a text report showing the main classification metrics, in this case it's usefull.

```
In [393]: print("Accuracy")
print(accuracy_score(y_test, predictionsRF))

Accuracy
0.7079207920792079
```

Accuracy classification score.

In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

```
In [395]: print("Kappa")
print(cohen_kappa_score(y_test, predictionsRF))

Kappa
0.4136189195394758
```

Cohen's kappa: a statistic that measures inter-annotator agreement.

```
In [369]: print("Confusion Matrix")
print(confusion_matrix(y_test, predictionsRF))

Confusion Matrix
[[169 130]
 [ 47 260]]

Compute confusion matrix to evaluate the accuracy of a classification.
```

But evenf or this model it's shown that the kappa value is not representative value to evaluate a good model, the same behavior is shown in the confusion matrix and in the classification report. Even the accuracy of this model is good value, but as overall summary this model can be improved with more tunning.

To predict a human behavior is very difficult, and in this project we choose to do classification if the song will be like it or not, but in future a recommendation is to use a regression problem in which you can give a percentage on how you will like a song.

5.4 Feature importance

After build an evaluates the 4 models that has been develop in the project, we analasys the data of "RF" which is the one that has been choosen. In this case we noticed that the Top five features that are:

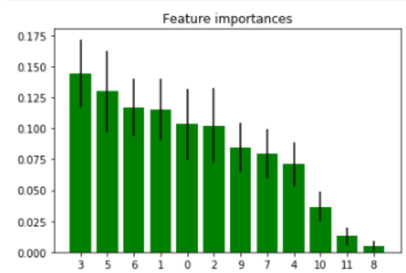
- **Instrumentalness:** Predicts whether a track contains no vocals
- **Loudness**: Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks.
- **speechiness**: Speechiness detects the presence of spoken words in a track.
- **danceability** : Describes how suitable a track is for dancing based on a combination.
- **acousticness** A confidence measure from 0.0 to 1.0 of whether the track is acoustic

```
In [380]: # Print the feature ranking
          print("Feature ranking:")

          for f in range(X_train.shape[1]):
              print("%d. feature %d: %s (%f)" % (f + 1, indices[f],
              spotifydataFrameFeaturesDF.columns[indices[f]], importancesRF[indices[f]]))

Feature ranking:
1. feature 3: instrumentalness (0.144294)
2. feature 5: loudness (0.129703)
3. feature 6: speechiness (0.116800)
4. feature 1: danceability (0.115098)
5. feature 0: acousticness (0.103291)
6. feature 2: energy (0.102188)
7. feature 9: valence (0.084448)
8. feature 7: tempo (0.079224)
9. feature 4: liveness (0.070851)
10. feature 10: key (0.036556)
11. feature 11: mode (0.012762)
12. feature 8: time_signature (0.004785)
```

```
In [381]: # Plot the feature importances of the forest
          plt.figure()
          plt.title("Feature importances")
          plt.bar(range(X_train.shape[1]), importancesRF[indices],
                 color="g", yerr=std[indices], align="center")
          plt.xticks(range(X_train.shape[1]), indices)
          plt.xlim([-1, X_train.shape[1]])
          plt.show()
```

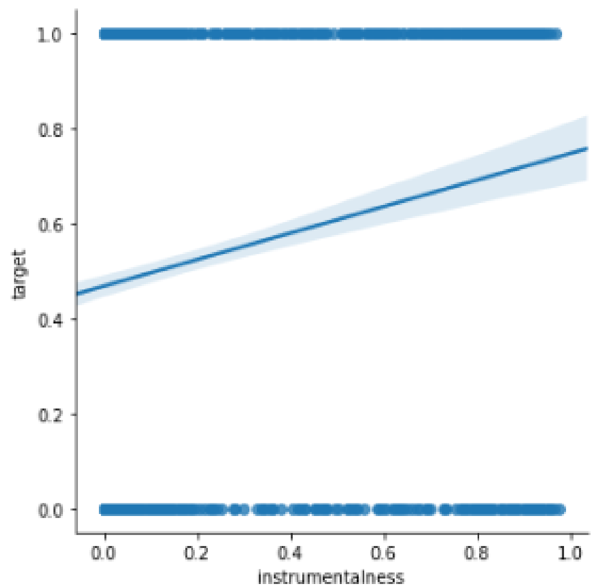


6. Plots base on the results

7. Anwers base in data

In this section we do quick summary base on vizualitation sto understand better the results the model that has bee choosen.

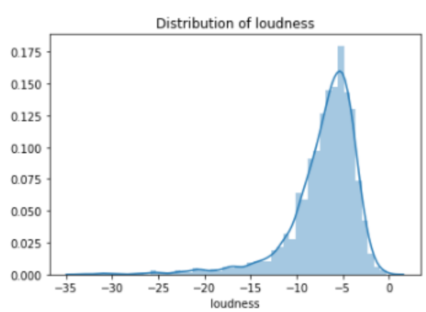
```
In [382]: sns.lmplot('instrumentalness', 'target', data=spotifydataFrame, palette='summer')
Out[382]: <seaborn.axisgrid.FacetGrid at 0x237d7970ec8>
```



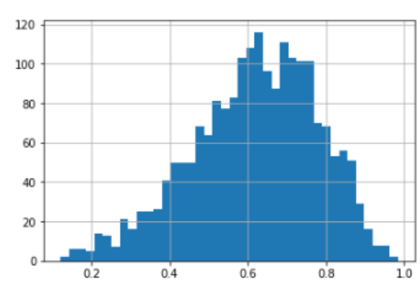
There is a linear relationship between the instrumetalness that is our first relevant feature, this means that if a track contains less vocal, will be better ranked.

It's important to review the distribution of the loudness and the danceability, this shoes that in the case of danceability this is distributed, but in the case of loudness there is tendency or bias to -5.

```
In [486]: instrumentalnessDis = spotifydataFrame['loudness'].dropna()
          # Distribution of age, with an overlay of a density plot
          ins_dist = sns.distplot(instrumentalnessDis)
          ins_dist.set_title("Distribution of loudness")
Out[486]: Text(0.5, 1.0, 'Distribution of loudness')
```

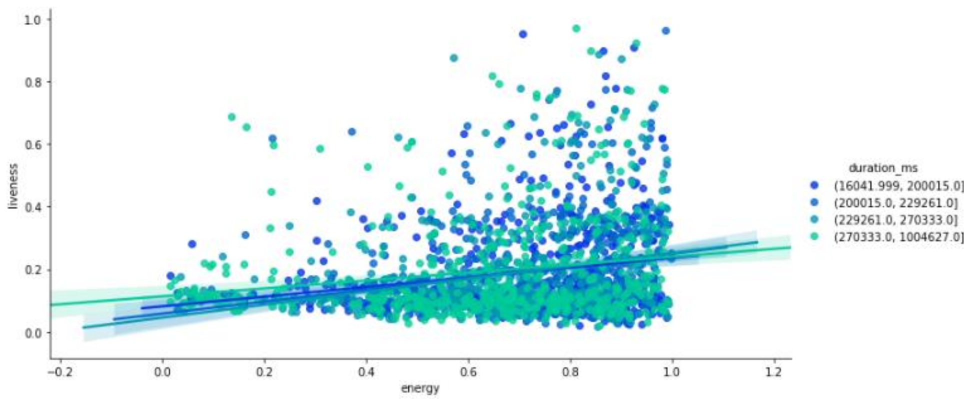



```
In [487]: spotifydataFrame['danceability'].hist(bins=40)
Out[487]: <matplotlib.axes._subplots.AxesSubplot at 0x237da7fbfc8>
```



In the EDA some features tthat has been choosen thinking can have a impact on the model like enery, or liveness are not that relevant as the previos.

```
In [438]: sns.lmplot('energy', 'liveness', data=spotifydataFrame, hue='duration_ms',palette='winter', aspect=2)#, hue_order=range(1,4))
Out[438]: <seaborn.axisgrid.FacetGrid at 0x237dc626548>
```



7 Recommendations

- The results of the models always ca be improved by more rounds of tunning, but this can be dangerous in order to overfit the model. So this step is very important to have experience and use the different evalutions best practices.
- The neural network model that has been used, the results shows that the performing of this model is very similar to regular machine learning algorithms, this is related of the size of the dataset that has been used.
- To predict if a song will be like it or not, base on the feature that Spotify provide is difficult due to intrinsit human behabior, but even tho there are some charactericties that plays esencial rol.
- Preprocess the data is Have to do activitiy, as same as feature enginnering to help the models to predict better.
- EDA has very usefull to have graphics and pictures on the data, that is very easy to explain to manameent or high executives of the company.
- The actual data set is relative small, due to only have aound 2000 observations, for better results can be benefical to have more data.
- The actual implementation is classification problem, in which the result is a song will be like or not, but for another project this can be implemented with different data set as regression problem, in which the algo give a idea how % you will oike the song.

8 Back up

8.1 Data set context

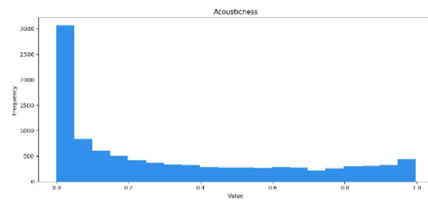
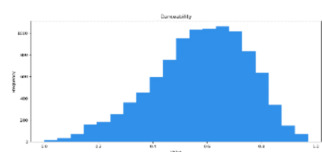
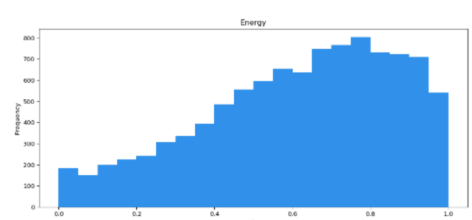
A dataset of 2017 songs with attributes from Spotify's API. Each song is labeled "1" meaning I like it and "0" for songs I don't like. I used this to data to see if I could build a classifier that could predict whether or not I would like a song.

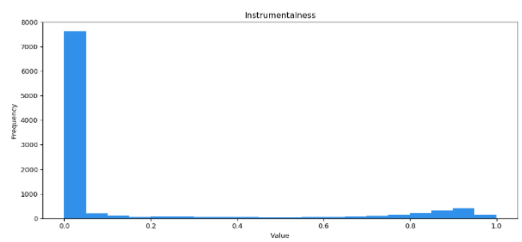
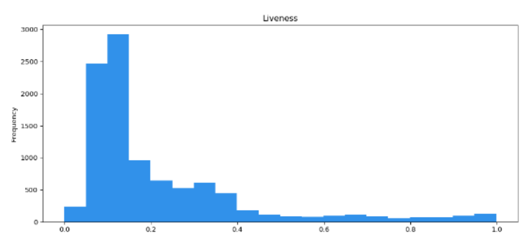
From →https://www.kaggle.com/geomack/spotifyclassification

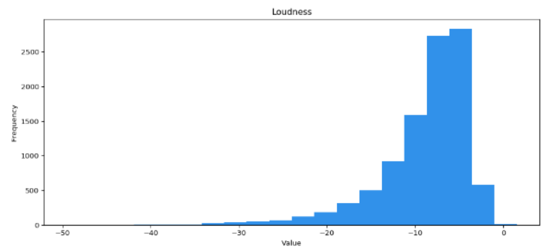
8.2 Audio Features Object

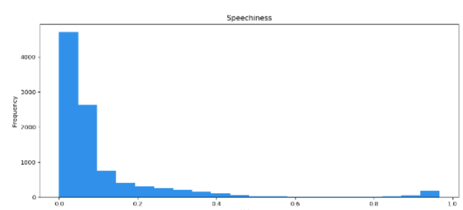
From → https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

KEY	VALUE TYPE	VALUE DESCRIPTION
duration_ms	int	The duration of the track in milliseconds.
key	int	The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation . E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.
mode	int	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
time_signature	int	An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).
acousticness	float	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic. The distribution of values for

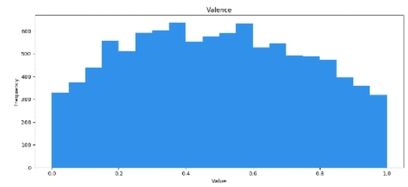
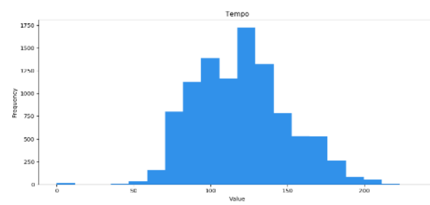
KEY	VALUE TYPE	VALUE DESCRIPTION
		this feature look like this: 
danceability	float	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable. The distribution of values for this feature look like this: 
energy	float	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy. The distribution of values for this feature look like this: 
instrumentalness	float	Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this

KEY	VALUE TYPE	VALUE DESCRIPTION
		context. Rap or spoken word tracks are clearly “vocal”. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0. The distribution of values for this feature look like this: 
liveness	float	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live. The distribution of values for this feature look like this: 
loudness	float	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db. The distribution of values for this feature look like this:

<table>
<tr><th>KEY</th><th>VALUE TYPE</th><th>VALUE DESCRIPTION</th></tr>
<tr><td></td><td></td><td>


</td></tr>
<tr><td>speechiness</td><td>float</td><td>Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks. The distribution of values for this feature look like this:


</td></tr>
<tr><td>valence</td><td>float</td><td>A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry). The distribution of values for this feature look like this:</td></tr>
</table>

KEY	VALUE TYPE	VALUE DESCRIPTION
		
tempo	float	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration. The distribution of values for this feature look like this: 
id	string	The Spotify ID for the track.
uri	string	The Spotify URI for the track.
track_href	string	A link to the Web API endpoint providing full details of the track.
analysis_url	string	An HTTP URL to access the full audio analysis of this track. An access token is required to access this data.
type	string	The object type: “audio_features”