International Journal of Music Science, Technology and Art
(IJMSTA) 5 (1): 39-51, Music Academy "Studio Musica", 2023 ISSN: 2612-2146 (Online)

Spotify Song Analysis by Statistical Machine Learning

Federica Biazzo[1]✉ and Matteo Farné[2]✉

[1]✉ Corresponding author: Federica.171@hotmail.it
[2]✉ Corresponding author: matteo.farne@unibo.it

ARTICLE INFO

Received: May 30, 2023
Accepted: June 16, 2023
Published: June 30, 2023
DOI: https://doi.org/10.48293/IJMSTA-97

Keywords:
Artificial Intelligence
Spotify
Machine Learning
Mood Playlists
Genre Playlists

ABSTRACT

This paper aims to study the use of Artificial Intelligence in the music streaming scenario. Specifically, the Spotify playlist creation process is redesigned by employing statistical machine learning algorithms. Spotify is chosen because it is the undisputed leader in music streaming, founding its success on the listening experience offered to its users. The level of personalization of the Spotify service is high and is made possible precisely because of Artificial Intelligence: by collecting masses of data about users and their listening habits, Spotify is able to understand the single user's interests and make targeted recommendations. Artificial Intelligence is also being used for another aspect: the creation of playlists, which collect different songs in order to satisfy every need or desire of the user. This work aims precisely at identifying the most relevant playlists by "Mood" through machine learning algorithms from a large sample of Spotify songs.

1 Introduction

Every day with our simple daily actions we generate huge amounts of data, something like millions of gigabytes per day. This massive amount of data is used by companies to obtain information about our behavior that can be useful for their business: for example, advertisements on websites that are based on our search history in order to offer products or services that we like; streaming apps possessing a recommendation system that uses the history of watched films or listened songs in order to collect information about what is of most interest to the customer and to offer suggestions; online shopping apps that recommend products based on the history, interests and preferences of the buyer.

All this is possible thanks to one of the most interesting new fields of research in science and engineering: Artificial Intelligence. It is defined as a system capable of learning, more precisely a set of instructions (algorithms) that allows computers to learn from experience (collected data) and, via this knowledge, to solve new problems in an ever-changing environment. A form of classification of AI which is based on the degree of intelligence allowed by the system has been identified: weak AI and strong AI [1].

The weak AI hypothesis is based on the idea that machines can act as if they were intelligent. A weak AI is a system that acts like the human mind: it collects data, studies them, works out solutions and chooses the most efficient one for the goal. In doing so, the machine "learns" to reduce the output error. Systems that are used to complete a specific task and require the presence of a supervisor to operate fall into this category.

The strong AI hypothesis, on the other hand, is based on the claim that machines that act intelligently actually think and do not merely simulate thought. Systems considered strong are programmed in a way that they develop their own intelligence and thinking mechanism, allowing them to learn from experience and to apply the same knowledge to different domains; they are therefore not limited like weak AI to completing a specific task. These types of systems do not need an expert to operate or to be fine-tuned, but they do have one major limitation: reaching such a level of intelligence requires a continuous and endless flow of data in the engine.

In addition to these two categories, the literature adds a third one: Superintelligent AI (ASI) [2]. The intelligences in this class

far exceed human intelligence; they are capable of scientific and creative thinking, possess common wisdom, social skills, and probably emotional intelligence. It is often assumed that this intelligence consists in a single super-computer, although it is more likely to be a network or a collection of different intelligences.
Among the streaming apps that base their operation on AI, we find Spotify. Spotify is the undisputed leader in music streaming and its success is due to the listening experience offered to its users: the availability of a large, varied and always up-to-date catalogue, the constant recommendations of new tracks based on the user's preferences and previous listening, the possibility of being able to create playlists and share them with friends [3]. By these services, Spotify has introduced an entirely personalized way of discovering music [4].
The personalisation process [5] starts with the user's registration, who is asked to self-identify their gender and age. Based on the IP address, the service detects the location, thus assigning a national identity. While using the service, all kinds of information about users, such as listening history, interactions with the user interface, but also the time of day, and whether they are using mobile or PC, are collected and routed to machine learning models that process them and make associations between different artists, tracks, podcasts, and playlists [6]. In this way, Spotify can offer increasingly targeted and intelligent recommendations.
The ultimate element of personalisation is the playlist: Spotify users can add tracks into shareable and collaborative playlists, thus incorporating a notion of sociality into the service and turning it into an increasingly personalised experience. Playlists are highly subjective: the same tracks can be included in playlists with different themes and the same playlists can also be included in different music categories.
The playlists that Spotify offers are divided into various categories to satisfy any need or desire of the user: by artist, by year, by musical genre, according to the location or situation in which the listener finds himself or herself. A particular category of playlists is Mood; this category brings together playlists related to moods: Cheerfulness, Thrill, Concentration, etc. This work aims precisely at redesigning the creation of Mood playlists on Spotify through machine learning algorithms. To this purpose, we employ a completely random sample of Spotify songs, and we derive the best clustering partition of sampled songs according to their audio features via the k-means algorithm. This leads to identify four data-driven labels describing the most representative moods in the sample. In the end, we also validate the informativeness of obtained clusters by four different mood prediction methods, namely, random forest, k-nearest neighbors, support vector machine and multilayer perceptron, all returning excellent results.
The paper proceeds as follows: Section 2 describes the rationale behind the four above mentioned methods, Section 3 presents the statistical analysis of a Spotify song dataset with a focus on the clustering partition obtained by the k-means algorithm, Section 4 reports the results of mood prediction obtained via those four methods, and Section 5 provides some concluding remarks.

2 Methods

In order to classify the songs collected in the sample according to their mood content, five machine learning algorithms were used. The k-means algorithm was used to identify the four clusters into which similar tracks are grouped according to their mood. Random forest, k-nearest neighbours, support vector machine and multilayer perceptron are the algorithms used to re-classify the tracks into their mood category, to assess the quality of the obtained partition. These methods fall into the category of weak AI [1]: they are supervised learning algorithms which use data to make predictions; the name is derived from the fact that there is a supervisor who can provide the desired output for any input based on the training data [7].

2.1 K-means

The k-means method (see [9], p. 460), also called McQueen's method, is an unsupervised learning algorithm: there is no supervisor, and no predefined outputs are provided; only the input data are available and the aim is to recognise patterns in the data and understand their structure.
Once the number of clusters k has been fixed, the k-means method randomly identifies k observations that will serve as initial cluster centres, called centroids; subsequently, for each centre the method identifies the subset of training points closest to it, which means that the cluster is identified. Once this initial partition has been made, the averages of each feature are calculated

for the data points in each cluster and this average vector becomes the new centroid of that cluster. This process is repeated until convergence.
During the cluster identification process, since the objective is to assign nearby points to the same cluster, a loss function is specified and an attempt is made to minimise it through a combinatorial optimisation algorithm. The loss function is:

$$W(C) = \frac{1}{2}\sum_{c=1}^{k}\sum_{i\in P(c)}\sum_{j\in P(c)} d(x_i, x_j),$$

where $P(c)$ denotes the c-th cluster, c=1,$\cdots, k$, and $d(x_i, x_j)$ is the Euclidean distance between points i and j. This function is defined as the "within clusters" deviance.
Similarly, a function can be defined that instead indicates the between-cluster deviance and tends to increase when observations assigned to different clusters are far apart:

$$B(C) = \frac{1}{2}\sum_{c=1}^{k}\sum_{i\in P(c)}\sum_{j\notin P(c)} d(x_i, x_j).$$

In order to search for the optimal clustering, the objective is to minimise W(C), which leads to a maximisation of B(C).

2.2 Random Forest

Random forest [8] is part of ensemble learning, that consists in the simultaneous use of two or more algorithms to make predictions [9]. In fact, when the outputs generated by several machine learning algorithms are combined, the result may be more accurate than the predictions of each individual algorithm. In the case of random forests, the sets of trained learners are decision trees [10]. A decision tree consists in the roots containing the entire dataset, the internal decision nodes and the terminal leaves. The idea behind this model is to search for the optimal decision sequence by performing a sequence of tests. In fact, starting from the roots, each node implements a test function, a decision rule that leads to the division into two child nodes (the node that generates them is called the parent), each one containing a non-overlapping subset of the input dataset. The process is repeated recursively up to the leaf nodes (nodes without children or terminal nodes) and stops when only pure leaves are obtained, i.e. containing points belonging to a single category [11].
The test function is first implemented on the most important attribute, i.e. the one that determines the largest impact on classification; from this, each test function subdivides the input space into smaller and smaller regions. Each leaf node has an output label, which in the case of classification is a class membership (classification tree) and in the case of regression is a numerical value (regression tree).
The structure of the tree is not fixed a priori, but varies during the learning process, adding branches and leaves as it goes along, depending on the complexity of the problem under consideration.
In the case of classification trees, the goodness of a split is quantified by the impurity, i.e., a measure of the uncertainty of a random variable: a split is pure if after the split, for all branches, all instances in every single branch belong to the same class. A high impurity indicates that points belonging to several classes are present in the node, while an impurity of 0 indicates that only one class is present. Since the division process continues until there is a pure leaf, to make the choice optimal it is necessary to evaluate each decision rule and select the one that minimises impurity. Two well-known impurity measures are the Gini heterogeneity index [12] and the cross entropy or Shannon entropy [12].

2.3 K-nearest neighbours

The k-nearest neighbours (k-NN) [9] is a supervised learning algorithm that can be used both for classification and regression: it consists in searching for the k training points that are closest in distance to the point x_0; in the case of classification, the plurality vote of the neighbours (which is the majority vote in binary classification) is taken as the predicted class, while in the case of regression, the mean or median of the k neighbours is taken or a linear regression problem on the neighbours is solved. The first step of this algorithm selects a distance measure between x_0 and the training points; distance is a measure of the degree of dissimilarity between two points and there are several types depending on the case under consideration: Minkowski distance, Euclidean distance, Manhattan, or city-block distance, Hamming distance, etc [10].
Having chosen the distance, k-NN arranges all distances in ascending order and searches for the k closest training points; the idea is that similar samples may share the same features. When a new set of training points is presented, the search is repeated with two possible variants:

- the k-nearest neighbours are calculated with a predefined value of k;

- all neighbours whose distance is less than or at most equal to a predefined radius r are calculated.

In cases where we have a lot of data and where the spaces are low-dimensional, the method performs well, which means that the probability to find enough nearby data points to get a good answer is high. On the other hand, in high-dimensional spaces the distances to the nearest neighbours explode, due to the curse of dimensionality: given M points, the algorithm would have to calculate all distances in pairs and the number of such distances would be equal to M^2. Since calculating a distance in an N-dimensional space requires N operations, the total complexity would become $O(M^2N)$ and this could only be reasonable for small values of M and N [12]. To solve this problem and reduce the computational complexity, some important strategies are employed, such as k-d trees and ball trees [12].

2.4 Support Vector Machine

Support Vector Machine (SVM) [9] is a discriminant-based method used in classification and linear regression. Considering the case of classification, given a set of training examples, SVM has the objective of searching for the separator that is as far as possible from the points labelled with class membership; this separator is called the maximum margin separator and is identified from the set of possible linear separators by searching for the separator ensuring the largest distance with the nearest observation [12]. The points closest to the separator constitute the support vectors and their distance from the separator is called the margin. The goal of the algorithm is to maximise the margin.

The separator is defined as the set of points $\{\boldsymbol{x}: \boldsymbol{w} \cdot \boldsymbol{x} + b = 0\}$ where **x** is the data vector, **w** is the weight vector, and b is a scalar parameter. A representation called dual representation is used to solve the problem: the optimal solution is obtained by solving the following problem:

$$\arg\max_{\alpha} \sum_j \alpha_j - \frac{1}{2}\sum_{j,k} \alpha_j \alpha_k y_j y_k (\boldsymbol{x}_j \cdot \boldsymbol{x}_k),$$

where y is the response vector, $\cdot$ denotes the inner product, the pairs $(j,k),\ j,k = 1,\dots,N$, identify all possible pairs of input observations, and α is a weight vector such that α_j are weights ensuring $\alpha_j \geq 0$ and $\sum_j \alpha_j y_j = 0$. The expression is convex and has a single global maximum that can be found efficiently. This is a quadratic programming optimization problem and can be solved using good software packages. Once the vector α has been found, we can get back to **w** by the equation $\boldsymbol{w} = \sum_j \alpha_j \boldsymbol{x}_j$ or we can stay in the dual representation [1]. Once the optimal vector α_j has been calculated, the equation for the separator $h(\boldsymbol{x})$ is as follows:

$$h(\boldsymbol{x}) = \text{sign}\left(\sum_j \alpha_j y_j (\boldsymbol{x} \cdot \boldsymbol{x}_j) - b\right).$$

2.5 Multilayer Perceptron

Deep Learning is a subfield of Artificial Intelligence and uses models called neural networks, inspired by the structure of the brain. Compared to traditional models, aiming to predict the values of the dependent variable and describe the relationships between variables, neural networks represent a black box model that knows how to process the input information to produce a good output but cannot say anything about the phenomenon under investigation [13]. A neural network can be used both in regression and classification problems.

A neural network [14] consists in a collection of interconnected artificial neurons, also called nodes or processors: some receive information from the environment, others emit responses in the environment and still others communicate only with units within the network. These units are called respectively:

- Input units X_p, which make up the input layer and transmit the observed values of a predictor to each hidden neuron;
- Hidden units Z_m, called in this way because they are not directly observable, which make up the hidden layer and which process the signals received and transmit their response to each output neuron;
- Output units Y_k which make up the output layer and which, like the hidden units, process the received signals and transmit their own response.

The distinction between input neurons and output neurons, the number of synapse layers (or neurons) and the connections constitute the architecture of a neural network. According to the type of architecture, two classes of networks are distinguished, hetero-associative and auto-associative [15].

In hetero-associative networks, the input nodes that receive input from the external environment are distinct from the output nodes that provide the network's response. This type of networks learns to associate pairs of different vectors, not necessarily composed of the same number of elements, the input vector and the response vector.

Auto-associative networks, on the other hand, have a single layer of fully connected units, so each unit receives input from both the external environment and the other units. The network's response is calculated by repeating the calculation of the activation of its nodes several times. Auto-associative networks are mainly used for pattern storage.
In the family of hetero-associative networks, two types of networks are distinguished according to the number of synapse layers: Single Layer Perceptron and Multilayer Perceptron [15].
The SLP (Single Layer Perceptron) is a type of network with only one layer of synapses; the basic idea of this network is that several artificial neurons work independently on the same inputs and use their respective outputs as input sources for one or more unconnected artificial neurons.
A perceptron with a single layer can only approximate linear input functions. In non-linear contexts, an MLP (Multilayer Perceptron) with internal neurons and more than one layer of synapses must be used. The response of a multilayer network is obtained by calculating the activation of one layer of neurons at a time proceeding gradually from the internal nodes towards the output nodes.
When the input neurons of the neural network receive a stimulus (input vector or pattern), these signals travel in parallel along the connections to the other units to which they are connected via synapses; these act as a filter that transforms the received message into an inhibitory or excitatory signal by increasing or decreasing its intensity. Each node only processes local information, which means it is only activated according to the information it receives, but it does not know what the general purpose of the processing is, or which operations are carried out by the other nodes to which it is not connected. The configuration of the connections and the values of the artificial synapses determine the behavior and response of the network.
The activation potential A of an i-th neuron is the algebraic sum of the products of all input signals x_j and the values of the corresponding synapse weights w_{ij}:

$$A_i = \sum_j^N w_{ij} x_j. \quad (5)$$

The response of neuron y_i is calculated by subjecting the activation potential to the action of an activation function $\Phi(A)$, which determines the type of response emitted by the neuron:

$$y_i = \Phi(A_i). \quad (6)$$

The activation function of a hidden layer allows a transformation of the received inputs which are transmitted to another hidden layer or an output layer [16]. This activation function can be:

- identity function.
- logistic sigmoid function.
- hyperbolic tangent function.
- rectified linear unit function.

The activation function of the output layer allows a final transformation of the output vector and must be chosen on the basis of the codomain of the dependent variable. Possible choices are:

- identity function (regression).
- logistic function (classification).
- softmax function (K-class classification).

A neural network learns to provide appropriate responses for each input stimulus by changing the values of its synaptic connections according to learning rules. Training a network means presenting it with a set of examples so that the network builds up its own internal knowledge necessary to perform the required task.

3 Application to a Spotify song dataset

3.1 Data exploration

The Python programming language was used to cluster music tracks by mood content and then classify songs according to obtained mood labels. We used the following libraries:

- NumPy is the fundamental package for scientific computing that provides multidimensional array object, various derived objects and an assortment of routines for fast operations [17].
- Pandas is an open-source library that provides high-performance data structures and data analysis tools [18].

- Seaborn is a library for making statistical graphics [19].
- Matplotlib is a library used for creating static, animated and interactive visualisations [20].

As the first step of the work, data collection was carried out [21]. Song audio features and other information are extracted from Spotify's Web API [22], a technology that allows access to metadata about music artists, albums and tracks directly from Spotify's data catalogue.

Once access has been authenticated, it is possible to extract all the necessary information from the catalogue for analysis purposes. As a first step, many tracks were randomly selected from the catalogue. For each of these, catalogue information and audio features were extracted. In Table 1 and in Table 2, those attributes are listed, and a brief description is given for each [22].

Table 1. Catalog Information from Spotify API Docs.

Catalog Information	Description
track.id	The Spotify ID for the track.
track.name	The name of the track.
track.artist	The name of the artist.
track.popularity	The popularity of the track. The value will be between 0 and 100, with 100 being the most popular.
track.album.id	The Spotify ID for the album.
track.album.name	The name of the album.
track.album.release_date	The date the album was first released.
playlist_name	The name of the playlist.
playlist_id	The Spotify ID for the playlist.
playlist_genre	The genre of the playlist.
playlist_subgenre	The subgenre of the playlist.

Table 2. Audio Features from Spotify API Docs.

Audio Feature	Description
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.
energy	Is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
loudness	The overall loudness of a track in decibels (dB). Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 dB.
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.
acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
instrumentalness	Predicts whether a track contains no vocals. The closer the instrumentalness value is to 1.0, the greater the likelihood the track contains no vocal content.
liveness	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.
valence	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).

duration_ms	The duration of the track in milliseconds.

3.2 Descriptive analysis

The listed features all assume values between 0 and 1, except loudness. We then proceed to normalise this variable, to make it independent of its measurement scale (decibels) and make it comparable with the other attributes.

Having collected the necessary information, an initial descriptive analysis [23] of the dataset was carried out (Fig. 1). The available dataset contains 33,042 observations and 20 variables. An analysis of some quantitative variables shows that most of the selected tracks are little played by users (average popularity = 40.92, peaking in the range 0-5), have a combination of musical elements that make them generally ideal for dancing (average danceability = 0.66), are in fact particularly energetic (average energy = 0.70) and not very acoustic (average acousticness = 0.16). The tracks turn out to consist mostly of music and few vocal and spoken word elements; demonstrating this, the averages of speechiness and instrumentalness are very low, almost close to 0, 0.11 and 0.09 respectively. An interesting feature of the dataset is that it is generally made up of tracks that convey positivity to the user, with an average valence of 0.51.

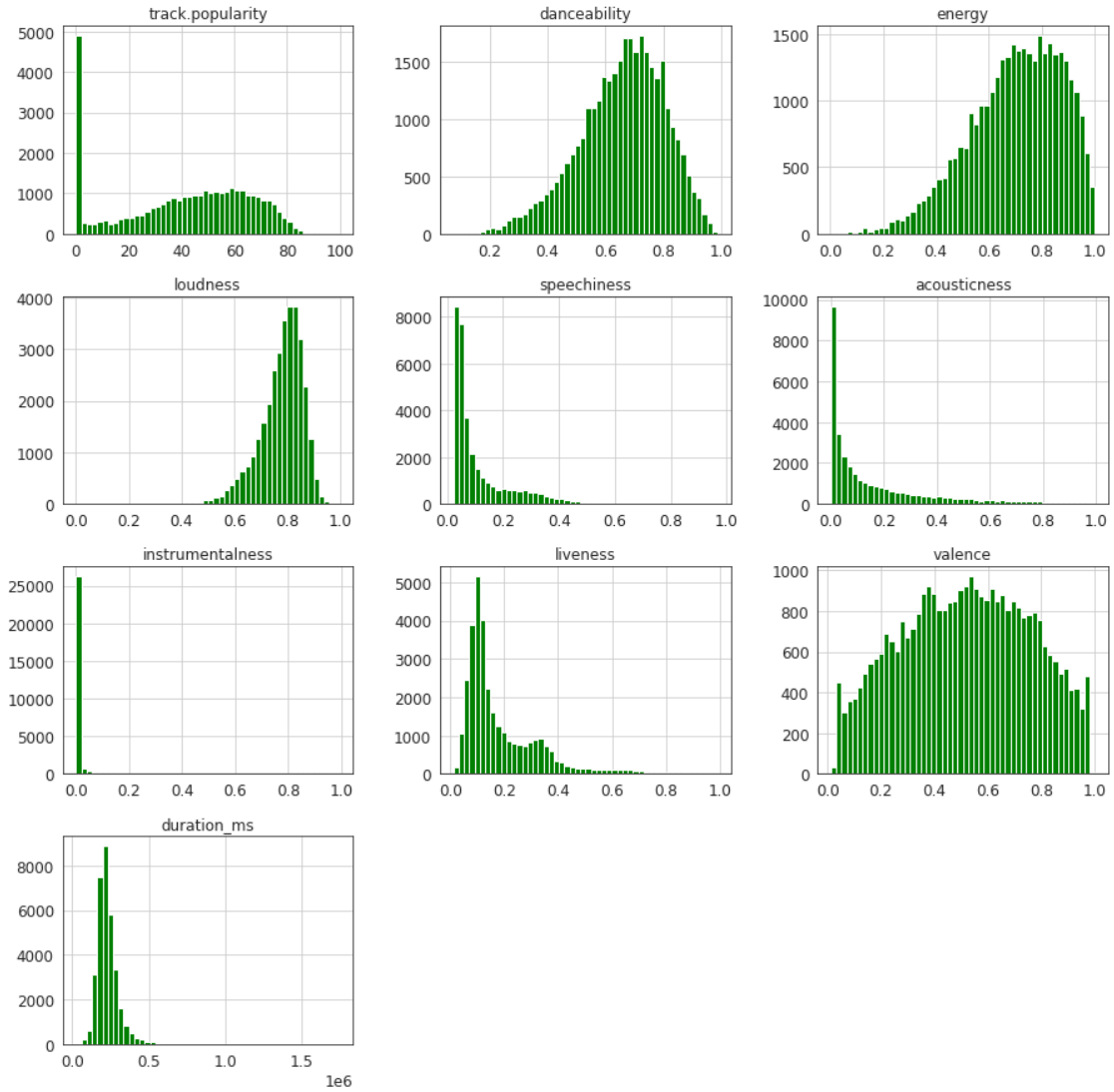

Fig. 1. Exploring Spotify Catalog Information and Audio Features: empirical distributions of *track.popularity* and the nine audio features described in **Table 2**.

A systematic exploration of the variables' distribution revealed the presence of some outliers for *duration_ms*: these are observations which deviate significantly from the values they take on in the rest of the observed units with respect to one or more variables. The presence of these values could distort any statistical analysis; consequently, they are eliminated. To this

end, the *boxplot* function is used; this displays the distribution of the variable in a compact manner: the width of the "box" represents the interquartile range, i.e. the difference between the third and the first quartile, and encloses the central 50 per cent of the observed values of the variable; the line that crosses the "box" is the median, while the segments that extend from it are called "whiskers" and indicate the dispersion of the values below the first quartile and above the third quartile. From the construction of the boxplot illustrated in Figure 2, the presence of outliers is confirmed, and they are eliminated: from the initial 33,042 tracks, the number of tracks is reduced to 32,963.

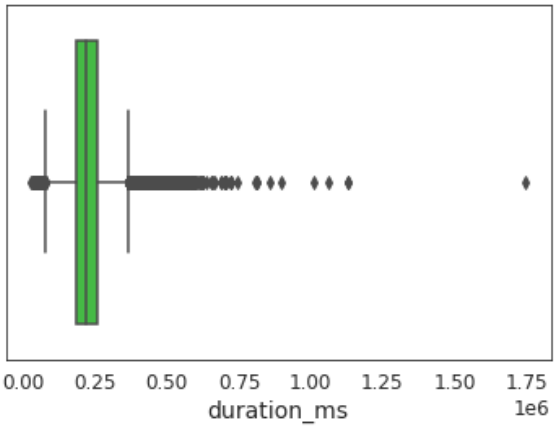

Fig. 2. Boxplot of *duration_ms* across the sampled 33,042 tracks.

Then, we examine the audio features in order to study their relationships. The correlation matrix represented in Figure 3 shows correlation levels close to 0 except for a few features: the highest correlation is recorded between *energy* and *loudness* (0.66). This result is not surprising, since among the perceptual characteristics that contribute to determine the level of energy in a track, in addition to dynamic range, timbre, onset rate and general entropy, there is perceived loudness.
Remarkable correlation indices are recorded between *valence* and *danceability* (positive correlation, tracks that convey positive feelings are more suitable for dancing), between *energy* and *acousticness*, between *loudness* and *acousticness* (negative correlations). Given its high correlation level with *energy*, *loudness* was removed.

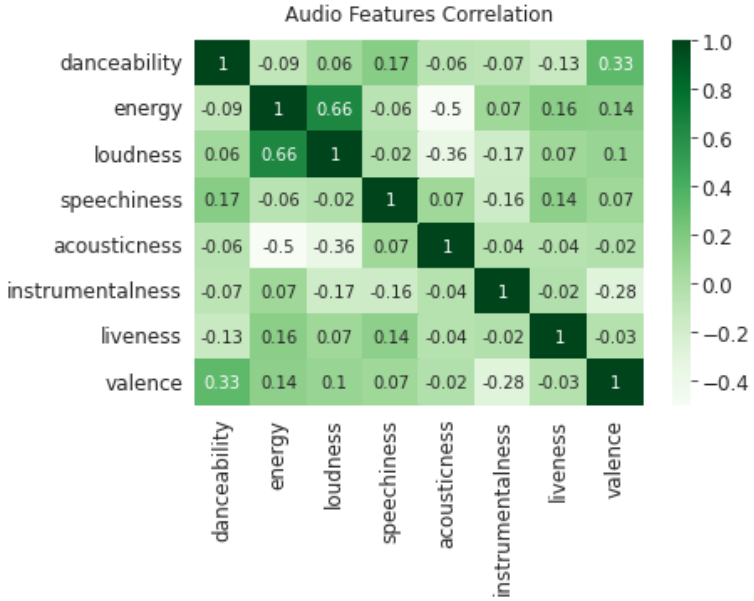

Fig. 3. Correlation Matrix of Audio Features.

3.3 Clustering songs by mood

Once prepared the dataset, the next objective is to group the different songs according to the audio features considered and to identify for each group the mood that these songs arouse in the user [24]. To do this, we resorted to clustering, which is an unsupervised learning task that searches to group together points that are similar to each other and different from the rest. Among clustering methods, the k-means algorithm (see [9], p. 460) was chosen.
To run the algorithm, it is necessary to provide it with the number of clusters into which to partition the dataset. To determine the optimal number, the elbow method [8] was used: the k-means algorithm is run for $k = 1, \dots, 15$ clusters and the within deviance is reported on a graph (Figure 4) for each value of k. The "elbow" is visually evident in correspondence of the value $k = 4$. The algorithm is then re-executed, setting four clusters as the optimal number.

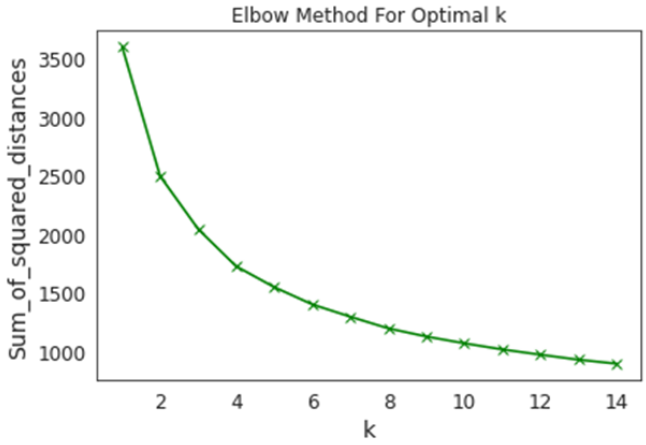

Fig. 4. Within deviance of k-means algorithm as *k* increases

Once the algorithm has been run, it is desired to make a graphical representation of the identified clusters. However, the number of variables considered in the analysis does not allow this, which is why data reduction techniques were applied to reduce the number of variables: principal component analysis (PCA) and t-distributed stochastic neighbour embedding (t-SNE).
The first method consists in replacing the variables in the model with a new reduced set of variables called factors or principal components. These attributes are orthogonal to each other and preserve the information content of the data, reducing its loss [8]-[25].
The second method consists in constructing a new low-dimensional space based on the original one; a neighbourhood statistic is calculated and then the points are placed in the new space making sure to keep the two spaces as close as possible [10]-[12].
The two principal components identified are linked to the other audio features by the relationships shown in Table 3.

Table 3. Relationship between Principal Components and Audio Features.

Principal Components	Audio Features				
	danceability	loudness	speechiness	acousticness	liveness
PC-1	-0.068434	-0.164926	0.041162	0.981982	-0.046191
PC-2	-0.558314	0.020922	0.033627	0.002175	0.828681

It appears from the graphs in Figure 5 that identified clusters are delineated with clear boundaries and some overlaps (especially between clusters 1 and 2). The largest cluster is Cluster 4 (13,646 tracks), which constitutes about 42% of the entire dataset; this is an indication that a predominant mood is present among the tracks in the sample. Clusters 3, 2 and 1 follow in size.

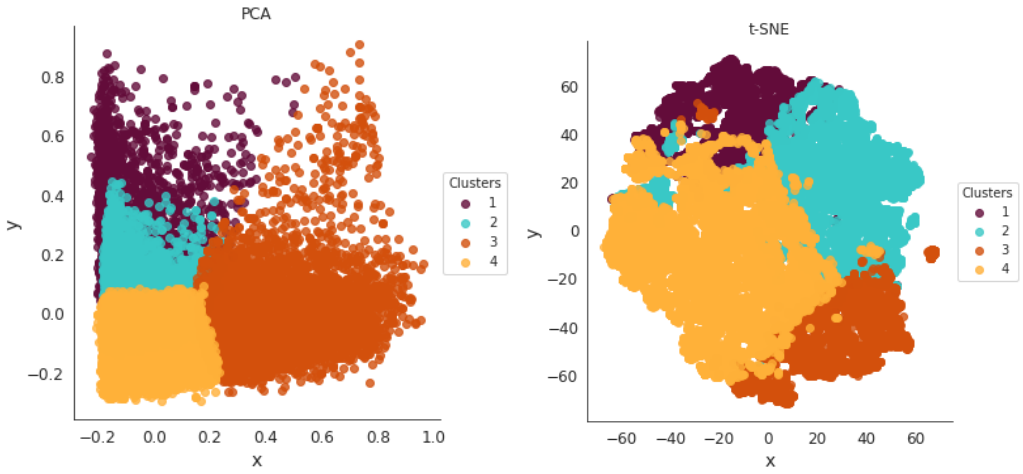

Fig. 5. Cluster Representations: (left panel) Principal Component Analysis, (right panel) t-distributed stochastic neighbour embedding.

Once formed the clusters, the next step is to study them, which means to analyse the type of songs and the values of the audio features characterising each cluster (Table 4) and then to assign to each of them a label identifying the mood associated with that particular set of songs.

Table 4. Audio Feature Values for each Cluster.

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
danceability	0.639876	0.521521	0.622395	0.766765
energy	0.780316	0.763192	0.538939	0.692904
speechiness	0.127498	0.075059	0.116330	0.129717
acousticness	0.082957	0.060257	0.575782	0.104418
liveness	0.459280	0.148153	0.168120	0.128294
valence	0.515888	0.434538	0.483035	0.579802

Cluster 1 counts 5,013 tracks and is the least numerous. It is characterised by high values of *energy* and *liveness*: the tracks in this cluster are therefore energetic, fast, and loud, with a rather high probability that they were performed live. They are tracks that convey positivity and are suitable for dancing. It was therefore decided to label this cluster "Energetic".
Cluster 2 consists of 5,214 tracks. As far as the audio features are concerned, these assume slightly lower values than in Cluster 1, but are nevertheless rather similar; this is confirmed by the overlap of the two clusters noted in the graphs of Figure 5. This cluster shows lower levels of *valence* and *danceability* than the other clusters, but in general maintains good values and the songs still convey positivity and are enjoyable. For this reason, Cluster 2 was assigned the label "Good vibes".
Cluster 3, consisting of 9,090 tracks, differs from the others in the high value assumed by *acousticness* (almost 0.60). The tracks comprising it are mostly performed with acoustic instruments, have a lower energy level than the others, but retain a good *valence* value. Since the tracks are therefore slower, less noisy, and generally positive, suitable for users wishing to relax, Cluster 3 was assigned the mood "Chill".
Cluster 4 is the most numerous (13,646 tracks). It differs from the others by the high values assumed by *danceability* and *valence*: it is the most cheerful and "danceable" cluster, consisting of energetic, fast and loud tracks. Due to these characteristics, the latter cluster was assigned the label "Cheerful".

4 Method prediction

Assigned the labels to the clusters, the next objective was to train a supervised learning algorithm in such a way that, given a song, it can accurately predict its mood.

To assess the predictive ability of the model, it may be appropriate to apply the hold-out method, which consists in dividing the sample into two disjointed subsets, the training set, and the test set. The training set is the subset on which the algorithm is trained and on which the model is created. Its constituent observations are selected from the initial dataset by simple random sampling without reintroduction, and a training set consisting of 80% of the observations was defined for this case. The test set, as can be deduced from its name, is the subset on which the model is tested and through which assessments of its performance are made. The test set consists of the remaining observations, i.e., 20% of the initial dataset.
The supervised algorithms used to perform the prediction are described in Section 2 and were trained with the following characteristics:

- the measure of impurity used to quantify the goodness of a split in the random forest is the Gini heterogeneity index (Section 2.1).
- in k-NN the predefined number of nearest neighbours is three and Minkowski distance was chosen as distance measure (Section 2.2).
- in Support Vector Machine a linear kernel was chosen to identify the linear separators (Section 2.3).
- the multilayer perceptron is composed of 100 hidden layers, the activation function of the hidden layers is the rectified linear unit function and since it is a K-class classification problem the activation function of the output layer is the softmax function (Section 2.4).

Confusion matrices were then constructed for each model, depicted in Figure 6. On the x-axis are the labels predicted by the classifier, on the y-axis the true labels [26].

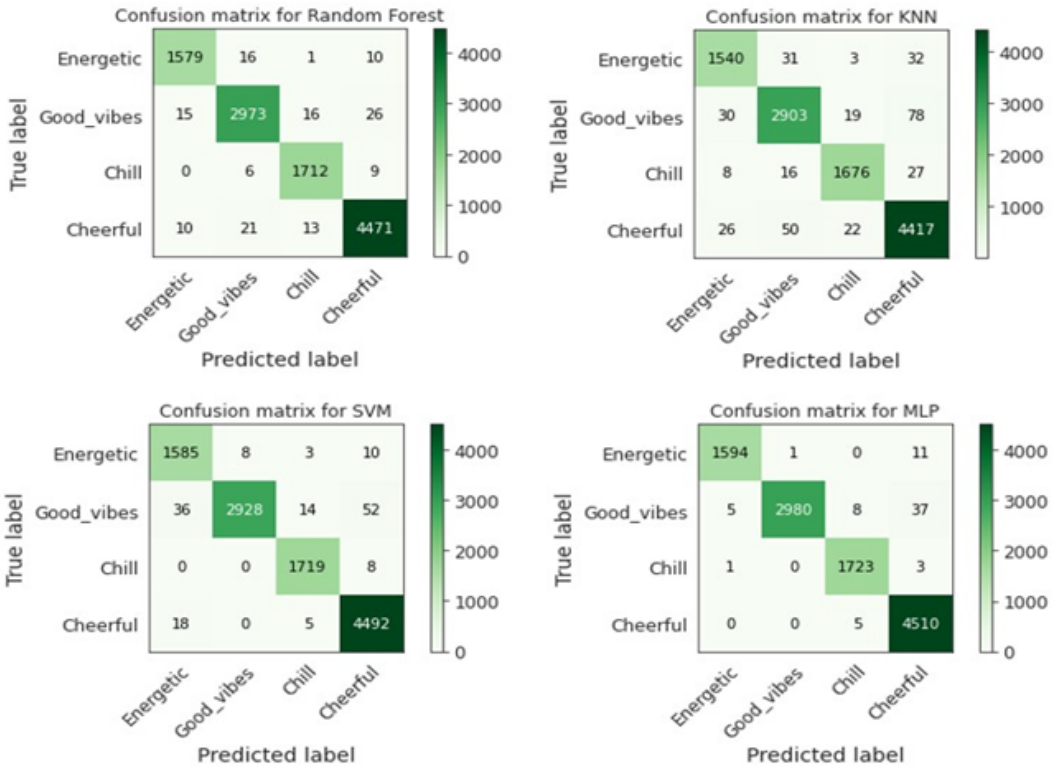

Fig. 6. Confusion Matrix for each model: Random Forest, k-NN, SVM and MLP.

From an initial view of the confusion matrices, it is possible to state that the label that was easiest to identify for all classifiers was “Chill”, i.e., the one relating to acoustic and less energetic tracks. When comparing the matrices, the classifier that showed the lowest error, i.e. the lowest number of false positives, was MLP followed by Random forest. On the other hand, the classifier with the highest number of false positives was k-NN.
From the matrix elements, it is possible to derive measures of classifier goodness, which are given below. These indices were calculated for each classifier used in this case (Table 5).

Table 5. Precision, recall and F1-score for each model.

	Random forest	k-NN	SVM	MLP
precision	0.99	0.97	0.98	0.99
recall	0.99	0.97	0.99	0.99
F1-score	0.99	0.97	0.98	0.99

As was evident from the comparison of confusion matrices, this latest analysis confirmed the accuracy of the MLP and random forest classifiers, which achieve precision, recall and F1 values extremely close to 1. The other two classifiers are equally accurate, but with slightly lower values. This proves that the mood labels derived in Section 3.3 are very informative and easy to predict, irrespectively of the used method.

5 Conclusion

In this paper, we have performed a k-means algorithm on a large sample of songs drawn by Spotify, and we have identified four clusters, representing the song moods. In particular, we have labelled the four clusters as “Energetic”, “Good vibes”, “Chill”, and “Cheerful”, according to each cluster’s features. In the end, we have performed a systematic prediction exercise of the obtained cluster labels via four different supervised statistical learning methods, namely, random forest, k-nearest neighbours, support vector machine, and multilayer perceptron. Prediction performance was surprisingly excellent for all methods, which also is a confirmation of our partition goodness. The described approach provides a statistically founded way to obtain valid song playlists according to their mood by the relative audio features.

References

[1] Alpaydin E. Introduction to machine learning. MIT press, 2020.
[2] Alpaydin E. Machine learning. MIT press, 2021.
[3] Breiman L. Random forests. Machine learning 45.1, 2001: 5-32.
[4] Bonaccorso G. Machine learning algorithms. Packt Publishing Ltd, 2017.
[5] Corea F. Artificial intelligence and exponential technologies: Business models evolution and new investment opportunities. Springer, 2017.
[6] Deng X., Liu Q., Deng Y., Mahadevan S. An improved method to construct basic probability assignment based on the confusion matrix for classification problem. Information Sciences, Volumes 340–34, 2016: 250-261.
[7] Eriksson M. et al. Spotify teardown: Inside the black box of streaming music. Mit Press, 2019.
[8] Floreano D., Mattiussi C. Manuale sulle reti neurali. Il Mulino, 2002.
[9] Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: data mining, inference and prediction. Springer, 2013 (corrected edition in 2017).
[10] Hull John C. Machine Learning in Business: An Introduction to the World of Data Science. Independently published, 2020.
[11] Hulaud S. Identification of taste attributes from an audio signal. U.S. Patent No. 9,934,785. 3 Apr. 2018.
[12] Kelleher John D. Deep learning. MIT press, 2019.
[13] Mazzetti A. Reti neurali artificiali – Introduzione ai principali modelli e simulazione su personal computer. Apogeo, 1991.
[14] Rajan K., Suh C., Mendez P.F. Principal component analysis and dimensional analysis as materials informatics tools to reduce dimensionality in materials science and engineering. Statistical Analysis and Data Mining: The ASA Data Science Journal, 1(6), 2009: 361-371.
[15] Russel Stuart J., Norvig Peter. Artificial Intelligence - A Modern Approach. Pearson, 2010.
[16] An introduction to seaborn, https://seaborn.pydata.org/introduction.html, last accessed 2022/06/09.

[17] How Spotify Uses ML to Create the Future of Personalization, https://engineering.atspotify.com/2021/12/how-spotify-uses-ml-to-create-the-future-of-personalization/, last accessed 2022/06/04.
[18] Matplotlib: Visualization with Python, https://matplotlib.org/, last accessed 2022/06/09.
[19] Numpy documentation, https://numpy.org/doc/stable/, last accessed 2022/06/09.
[20] Pandas documentation, https://pandas.pydata.org/docs/, last accessed 2022/06/09.
[21] GitHub, Spotify Emptions Project, https://github.com/SylCard/Spotify-Emotions-Project, last accessed 2022/04/30.
[22] Predicting my mood using my Spotify data, https://towardsdatascience.com/predicting-my-mood-using-my-spotify-data-2e898add122a, last accessed 2022/04/30.
[23] Scikit-learn-MLP Classifier, https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, last accessed 2022/06/06.
[24] Spotify: il valore unico di una proposta vincente, https://4books.com/it/approfondimento/spotify-il-valore-unico-di-una-proposta-vincente#chapter-8, last accessed 2022/06/04.
[25] Spotifyr, https://www.rdocumentation.org/packages/spotifyr/versions/2.1.1, last accessed 2022/06/06.
[26] Spotify Developers, Web API, https://developer.spotify.com/documentation/web-api/, last accessed 2022//04/29.