FOM Hochschule für Oekonomie & Management

Hochschulzentrum Düsseldorf

Scientific Paper

part-time degree program

5th Semester

in the study course "Wirtschaftsinformatik"

as part of the course

Big Data & Data Science

on the subject

Predicting Music Genres based on Spotify Song Data using a Gradient Boosting Algorithm

by

Thomas Keiser

Martin Krüger

Jesper Wesemann

Luis Pflamminger

Advisor: Prof. Dr. Adem Alparslan

Matriculation Number: 123456 (Krüger), 123456 (Keiser),123456 (Wesemann), 123456 (Pflamminger)

Submission: January 31st, 2022

Contents

List of Figures

List of Tables

List of Abbreviations

API   Application Programming Interface

REST   Representational State Transfer

List of Symbols

1 Einleitung

Dies soll eine LATEX-Vorlage für den persönlichen Gebrauch werden. Sie hat weder einen Anspruch auf Richtigkeit, noch auf Vollständigkeit. Die Quellen liegen auf Github zur allgemeinen Verwendung. Verbesserungen sind jederzeit willkommen.

1.1 Problem Definition

1.2 Goal

1.3 Structure

2 Fundamentals

2.1 Basic Concepts of Big Data

Big Data is an umbrella term used to describe various technological but also organizational developments. Originally, Big Data refers to large sets of structured and unstructured data which must be stored and processed to gain business value. Today, Big Data is also often used as buzzword to outline countless modern use cases that deal with large amounts of data [1, p.5]. Big Data is therefore often used in conjunction with other keywords like automatization, personalization or monitoring. This chapter presents the foundation of Big Data and gives an overview of technological and business standards. (QUELLE)

2.1.1 Relevance of Data

Data in combination with Business Intelligence has become increasingly important over the past decades and is closely associated with the advances of the internet itself [2, p.1165]. Looking back, Business Intelligence can be divided into three sub-categories, which follow another linearly. The first phase is centered around getting critical insights into operations from structured data gathered while running the business and interacting with customers. Examples would be transactions and sales. The second phase focuses increasingly on data mining and gathering customer-specific data. These insights can be used to identify customer needs, opinions and interests. The third phase, often referred as Big Data, enhances the focus set in phase two by more features and much deeper analysis possibilities. It allows organizations and researchers to gain critical information such as location, person, context often through mobile and sensor-based context [2, p.1166].

In conclusion, organizations require Business Intelligence as it allows them to gain crucial insights which is needed to run the business and achieve an advantage over the competition. It is important to minimize the uncertainty of decisions and maximize the knowledge about the opportunity costs and derive their intended impacts. It is clearly noticeable that the insights and analysis possibilities become progressively deeper and much more detailed. Along this trend the amount of data required becomes larger and larger with increasingly complex data structures. Size, complexity of data and deep analysis form the foundation of Big Data and can be found again in the 5V matrix of Big Data.

2.1.2 The 5V Matrix for Big Data

When describing Data, a reference is often made to the five Vs, which highlight its main characteristics. The previous aspects of Big Data can again be recognized in averted form.

Volume: The size of the datasets is in the range of tera- and zettabyte. This massive volume is not only a challenge for storing but also extracting relevant information [1, p.6].

Variety: Variety refers to the diversity of the data itself. For modern Business Analytics almost every data format and type plays a vital role. They range from the more classical text and figures to images, audio and video [1, p.6]. Whereas classical formats typically are stored in a structured way other formats rely on a semi- or unstructured database. The main differences between structured and unstructured data will be discussed in the following chapter as part of the different storage solutions for Big Data. Without much preface, unstructured data is more difficult to classify and further complicates the extraction of information but allows much deeper analysis possibilities [3, p.2f].

Velocity: Velocity is about the speed in which the data must be stored, and valuable information extracted. In a fast-paced environment, like the current globalized world, faster analysis can be a key advantage. Some special use cases, like malfunction detection, even require real-time processing of data [4, p.6].

Value: The goal of Business Analytics and the extraction of information out of data is, as mentioned already in the previous, to create business value (Big Data Analytics, 6). Value that minimizes uncertainty of action or processes and gives the operators a key advantage [1, p.6].

Veracity: Veracity describes a challenge of analytics with data. The gathered data is often vague and not concise. Its information is not easily identifiable from the outside. Furthermore, some samples of the dataset are often of bad quality for multiple possible reasons and therefore hinder the algorithm and it's training rather than supporting it. The predictions on the other hand must be precise. This conflict is a massive challenge when creating analysis models [4, p.6].

2.2 Differencation between Big Data and Big Data Analytics

The term Big Data Analytics is used in literature to describe the subcategory of Big Data that focusses primarily on the analytics of existing data, as the name already suggests. The analysis of data is of very high interest for many use-cases and organizations as its outcome is business value (comparison to the previous chapter: Why is data so important).

The term Analytics is used to describe a systematical analysis of data in some form. Analytics is about detecting hidden patterns, clusters and meaningful features ranging from simple detection over deep analysis and predictions. The goal is to process the data in such a way that value is created from it [3, p.2] [1, p.8f]. The creation of value is closely linked to the use cases for which in-depth knowledge is required and in which form. Generally, literature distinguishes between four main groups of analytics by grouping them according to the methodology and their objectives starting with Descriptive Analytics. Descriptive Analytics is the simplest form of Analytics and is often the first step for further research. Descriptive Analytics gives information about what has happened in the past. Descriptive Analytics is often used for reporting on relevant topics for operation such as KPIs, sales or revenue. Typically, visual presentation methods are used for displaying the information. Diagnostic Analytics is based on Descriptive Analytics but serves a different goal, since its goal is to give insights on why something has happened in the past. Diagnostic Analytics evaluates the impact of features, detects correlations and dependencies. Predictive Analytics takes this approach one step further as it predicts what is most likely going to happen in the future based on the knowledge gained from past data. It creates models with the help of algorithms to determine the probability of outcomes. The final step of Analytics is Prescriptive Analytics, which again is based on its predecessor. Prescriptive Analytics predicts outcomes and recommends actions to avoid or support them respectively [1, p.8f].

This project contains elements of descriptive, diagnostics and predictive analytics. Using the Crisp Dm process model (CHAPTER X), the data set is first analyzed and then a predictive model is generated to classify new data based on known data.

https://www.sigmacomputing.com/blog/descriptive-predictive-prescriptive-and-diagnostic-analytics-a-quick-guide/

2.3 Data Formats

As part of Big Data, the storage of data faces similar challenges as Big Data itself. Adequate storage solutions are key to providing data for the following analysis step. This chapter gives a quick overview on data types. Storage is a very important topic for Big Data but plays only a minor role for the project itself and therefore will not be discussed in more detail.

Variety already outlined the shift from structured to unstructured data formats. Structured data has a fixed format and fits into a predefined data model which can be stored in tabular form. Unstructured data, on the other hand, has no fixed format, schema or structure.

It comes in almost every form such as PDF, text, image, audio and many more. Basically, the whole internet and everything that is published on it is some form of unstructured data. Therefore, it is believed, that approximately 95 percent of all data is in unstructured form. In between structured and unstructured data there exists a subcategory called semi-structured data. Semi-structured data has no strict standard and but can be read by machines since it often consists out of user-defined data tags. An example for semi-structured data is XML [3, p.2f].

Structured and semi-structured data is relatively easy to analyze compared to unstructured data because machines mostly rely on structural organization. Unstructured data, on the contrary, has great potential because the amount of information stored inside it is huge. To analyze unstructured data, it is often required to deconstruct it into metadata which again is comparable to semi-structured data. Often both structured and unstructured data is necessary to form well-founded business decisions and gain a competitive advantage [3, p.2f].

The dataset used for this project and described in more detail in chapter X is a structured dataset. It consists of classical numerical and categorical features which all have a predefined range of values.

2.3.1 Reinforcement Learning

2.3.2 Categories of Machine Learning

Machine learning refers to the ability of a computer to learn on its own. The field of machine learning includes a wide range of algorithms that learn from data and make predictions with variable quality. These predictions are not made programmatically but by data-driven predictions that are "learned" by generating knowledge. Basically, Machine Learning can be divided into 3 different methods which are discussed below.[5, p. 4]

Supervised Learning In supervised learning, a function is determined by mapping input values to known target values using examples. Supervised learning is also called learning from examples for this reason.[6, p. 96] The system learns based on a training data set that already contains the correct answers. Using the already given data sets, the algorithm learns to set up rules and patterns to reach the known target variable. The process is repeated until the prediction matches the desired quality. Experiences from each iteration are in turn included in the learning process. If the trained model fulfills the desired results, it can also be applied to unknown data. The goal of this learning method is therefore to make predictions and recommendations.[6, p. 96] It is also important to mention that

target values in machine learning are called labels. Here again, one must distinguish between two use cases. If the label is nominal, it is a so-called classification. So, the model should assign data to specific classes. A regression is present if the label has metric target values.[7, p. 46] The aim regression is to use data to make forecasts of future values or to identify trends. Problems that can occur when using supervised learning and should be avoided if possible are either overfitting or underfitting. Overfitting is when the algorithm is adapted too much to the training data. This means that it delivers better results when applied to training data than to unknown data. The opposite is called underfitting which means the models of the learning procedure are not complex enough. Therefore, the algorithm cannot deliver sufficient performance to make predictions.[8]

Unsupervised Learning The opposite of supervised learning is unsupervised learning. Here, the algorithm itself acquires patterns and correlations without explicitly predefining target values. Which is why the algorithm is also called learning from observations.[6, p. 97] This is also done using sample data, but without labeled output data as in supervised learning. The algorithm thus focuses on deriving or recognizing common structures and patterns in the sample data.[5, p. 7] The search includes the complete training data and not only relations with concrete target values as in supervised learning.[9, p. 802] Unsupervised learning can again be divided into several types. Clustering, for example, deals with finding frequency structures, patterns and grouping the data into a few sets.[10, p. 260] Another type of learning are associations which search for rules that map connections between data points. Finally, there is also dimensionality reduction. This kind of learning tries to reduce all available variables to the most important ones.[11, p. 10] The problem with unsupervised learning, however, is that no conclusions can be drawn about the quality of the algorithm due to the non-existence of target variables. Thus, there is no real right or wrong in the learning process and so the algorithm has the possibility to fail completely.[6, p. 97]

Semi Supervised Learning Semi-supervised learning is a learning method that includes elements of supervised and unsupervised learning. Because of this combination, it is often not considered a separate form of learning, but it is important to mention it. This method uses training data that are only partially labeled.[6, p. 98] This training data serves as a representative label of discovered structures if unsupervised learning is used. If supervised methods are used, the unlabeled data serves to make statistical accumulations more estimable. The main advantage of semi-supervised learning is that training of algorithms is already possible with little data. This reduces the high costs and the effort that often have to be spent on target values.[12]

Reinforcement Learning Reinforcement learning is fundamentally different from the learning methods already mentioned. Unlike the learning methods already discussed, no training data is available at the beginning.[10, p. 351] The algorithm acquires data only by interacting with the environment, which is why this learning method can also be called learning by interaction. Due to the few Requirements, this learning method is optimally suited[6, p. 98] Models train by tackling challenging problems and having their decisions immediately rewarded if successful or punished if unsuccessful through feedback signals. This reward or punishment occurs, for example, in chess when a game is won or lost. Moves that lead to victory are saved, as are moves that lead to a possible loss.[6, p. 98] In contrast to other learning methods, the algorithm does not know whether a decision is right or wrong before it decides what to do next. In addition, it does not know whether the decision it is currently making is the best one for the situation at hand, since its wealth of experience only grows with increasing runtime. In order to cover all possible situations, the algorithm must initially also cover previously unknown possibilities and not only act on the basis of actions that have led to reward or punishment in the past. However, if the algorithm is trained well enough, it can solve problems very well on this basis. Overall goal is to develop a strategy to maximize rewards received and get better as fast as possible.[10, p. 351]

2.4 Decision Trees

Decision Trees are one of the most widely used supervised Machine Learning Algorithms either as standalone solutions or in combination with enhancement approaches like Boosting. They are furthermore very flexible in their construction and can be used for various Machine Learning Problems such as Classification and Regression.

Decision Trees "predict an unknown value of a target variable by learning decision rules from data features" to reconstruct the dependence between the features and the respective labels for each sample. To perform Classification or Regression, Decision Trees rely on recursive splitting of the dataset into multiple subgroups. As the number of iterations increases, the subgroups become more and more homogeneous [13, p.330]. The ideal result is that each subgroup is fully homogeneous and therefore only represents a single category (in case of Classification). However, this is often only a theoretical best condition, as multiple risks, such as overfitting, are associated with the increasing depth of Decision Trees.

Trees consist out of four main components. A Node is a discrete Decision Function that takes samples as its input and splits them based on features into subgroups. The aim

of each split, as previously discussed, is to create a split that results in the overall most homogeneous distribution for all subgroups [14, p.6]. Nodes can be subclassified into three kinds. The Top-Node, from which the classification starts, is called a Root Node. Nodes that are located at the very end of a Decision Tree are referred to as Leaves. Leaves do not split data any further and only mark the end of a Decision Tree. When reached, Leaves categorize or predict a final output value depending on the prediction task. Nodes in-between the Root Node and Leaves are called Internal Leaves. Like the Root, Internal Leaves are responsible for the recursive splitting of the data. Branches connect Nodes with another. For classical Trees, information only flows from the top to the bottom of the Tree.

2.4.1 Decision Tree Algorithm

In practice, there exist various Algorithms for computing Decision Trees with the most common ones being: ID3, C4.5, C5.0 and CART. Each algorithm follows the same principle of regressively finding perfect splits to separate data but utilizes different methods to find the ideal splitting criteria, which strongly influences the structure of the Tree, its accuracy and performance. Additionally, each Algorithm has its benefits and constraints. Therefore, it is important to determine the best Algorithm before implementing a Decision Tree based on the prediction task and dataset. This project uses the Python library Sklearn to implement a Classification Tree. Sklearn is based on the CART Algorithm **sklearn Decision Trees**.

To better visualize the procedure of the Decision Tree Algorithm, a simplified dataset is used, on which the individual steps are explained. For this example, a Classification Problem is chosen. The dataset consists out of actual features and labels from the project implementation phase. The features are "acousticness" and "danceability". Both features are numerical with a value range in-between 0 and 0. The Classification Problem is binary with "Hiphop" and "Jazz" representing the classes k for which the samples of the dataset are classified. Mathematically, the dataset is represented in the following form: x_i present the explanatory features while y_i represents the corresponding label for one data point of the input dataset N with a total number of n samples.

(TABLE WITH DATA)

(COORDINATE SYSTEM WITH DATA)

With the initialization of the dataset complete, the implementation of the Decision Tree Algorithm can begin. The Decision Tree Algorithm splits a Node represented by Q_m with N_m samples into multiple subgroups. The split can be binary, which means that Q_m is divided

into two subgroups Q_m^{left} and Q_m^{right}, or multiway. While multiway splitting seems to be more advanced with greater prediction potential, in practice and for CART binary splitting is used almost exclusively (QUELLE). The split $\theta = (j, t_m)$ consists out of a feature j and a threshold t_m on which the division takes place. The Output $G(Q_m, \theta)$ is mathematically defined as the following **sklearn Decision Trees**:

$$G(Q_m, \theta) = \frac{N_m^{left}}{N_m} H(Q_m^{left}, \theta) + \frac{N_m^{right}}{N_m} H(Q_m^{right}, \theta)$$

The quality of the split is defined using an Impurity Function H. The Impurity Function has one Leaf Q_m^{left} or Q_m^{right} and the spliting criteria θ as its input. The best overall Gain $G(Q_m, \theta)$ is reached, if $G(Q_m, \theta)$ is minimized (3) **sklearn Decision Trees**.

$$\theta^* = argmin_\theta G(Q_m, \theta)$$

The most common Impurity Function H for Classification is Gini Index (4) which is also used for CART Decision Trees. Gini index measures the probability that a sample does not belong to the category that represents the majority of the subgroup [13, p.335]. If both categories of a subgroup are identical in size, the Gini Index reaches its maximum point at $0,5$ (5). The maximum of the Gini index means the worst possible data constellation for a subgroup. Gini index equal to 0 on the other hand represents the best possible result with the subgroup being fully homogenous. p_i represents that probability that a sample belongs to the class j.

(4) $Gini = 1 - \sum_{j=1}^{k}(p_j)^2$

(5) FORMULA AS A GRAPH

For the example the splits look like the following. The split of numerical features is more complicated as it is for categorical or binary features since it can take place at any value within the value range. Therefore, each value of the sample could be a possible threshold. To achieve the best overall Gini Index, it must be calculated for every possible threshold of every feature. The calculation below only shows the best possible splits for each feature according Gini Indexes. With G calculated for both features, the first split can be determined. When comparing both features it is visible that Danceability minimizes G more than Acousticness does and therefore is according to (3) determined as the overall best possible split for the Root Node.

Acousticness:

$Gini^{left} = 1 - ((\frac{2}{5})^2 + (\frac{3}{5})^2) = 0,48$

$Gini^{right} = 1 - ((\frac{3}{3})^2 + (\frac{0}{3})^2) = 0$

$$G(Q_m, \theta) = \frac{5}{8} * 0,48 + \frac{3}{8} * 0 = 0,30$$

Danceability:

$$Gini^{left} = 1 - ((\frac{4}{4})^2 + (\frac{0}{4})^2) = 0$$

$$Gini^{right} = 1 - ((\frac{1}{4})^2 + (\frac{3}{4})^2) = 0,36$$

$$G(Q_m, \theta) = \frac{4}{8} * 0 + \frac{4}{8} * 0,36 = 0,18$$

The splitting process is repeated for each subgroup until a stop-criteria is met. The natural stop criterion is a completely homogeneous dataset for a node and can also be found in the example for Leaf X. Such Internal Nodes automatically become Leaves and represent the category of samples. Other stop criteria can be predefined depending on the Algorithm used for implementation [14, p.7]. The most relevant criterion is the definition of a maximum depth of the Decision Tree. Other hyperparameters are discussed in the implementation chapter.

Tree optimization plays a very relevant role because Trees often overclassify training data without countermeasures [14, p.7]. Although the training data is well classified, overfitted Decision Trees often produce very poor results for test data. Too accurate classification of training data can negatively affect Decision Trees, as they are less able to generalize the learned knowledge. Pruning is a technique used to overcome overfitting problems by reducing the size of Decision Trees [13, p.331]. Sections that provide little to no classification benefit are removed or not constructed during the recursive splitting process. In essence, worse results for training data are traded for better results for unknown data.

The experimental dataset is not large enough to take appropriate countermeasures. The complete Classification Tree is shown in figure X. For the second iteration, only the left subgroup was further split. The result are two fully homogeneous sub-subgroups created from the data Nm of the left subgroup. Because only two features were used, each split can be visualized using a coordinate system (figure X2). the colored areas present the respective splits.

2.4.2 Evaluation of Decision Trees

In conclusion, Decision Trees can be assessed as follows. Starting with the advantages, the main benefit is the overall simplicity of Decision Trees, both from a technical and business point of view. For researchers and developers, Trees are easy to construct, require little to no data preparation, are almost univerally applicable with a possibility of validation.

However, the simplicity for business should not be underestimated either. When comparing Machine Learning Algorithms, the main comparison is often the accuracy of a model. The areas in which Decision Trees stand out include visualization and comprehensibility. The Decision Tree Algorithm is a white box model that allows complete transparency and explainability [13, p.339] **sklearn Decision Trees**.

The disadvantages of Decision Trees are again closely related to its simplicity. Overfitting and the relative instability of Decision Trees are the main drawbacks and result in good momorization but a comparatively weak generalization ability [13, p.339] **sklearn Decision Trees**.

CROSS VALIDATION: Cross Validation is a method structure the dataset for modelling. It allows the use of one single dataset for both training and validation as the dataset is randomly split into N sections. Each section contains an equal distribution of label data as the original dataset. One subset is reserved as a validation dataset while the other subsets are used for the modelling of the dataset. The modelling takes place N times and for every iteration another subset is used for validation. Therefore N models are created in total. Each model can agian be tested against the subset that is reserved for validation with the best model chosen [14, p.8f].

2.5 Gradient Boosting

The Gradient Boosting Algorithm is derived from Gradient Boosting Machines, which are a family of powerful Machine Learning Algorithms with a certain procedure pattern for the creation of models. In general, GBMs are very flexible in their characteristics with the possibility of utilizing multiple different Machine Learning Algorithms as their foundation.

Boosting differs from classical approaches as it does not consist out of a single predictive model but an ensemble approach. Ensemble Algorithms contain multiple Weak Learners that form a committee to create a strong prediction. Weak Learners are often very simple forms of traditional Algorithms, like Decision Trees, and must just be able to predict parts of the dataset correctly. Only the combination of many Weak Learners allows the model to perform overall accurate predictions. The most common form of Ensemble Algorithms are Bagging Algorithms with Random Forests as an example. Bagging, in essence, is the combination of multiple unique models. The prediction is formed by aggregating the outputs from all models into a single representative value. Typically, all models are derived from a single Algorithm, like Decision Trees for Random Forests, but technically there is no limitation to aggregate outputs from different Algorithms. This is also the case for Boosting.

Boosting, on the other hand, follows a different principle and does not rely on independent models with an aggregation function. Boosting fits new models sequentially and can thereby use earlier acquired knowledge for further iterations. This allows GBMs to train specific areas of the dataset where it has previously performed poorly [13, p.345f].

2.5.1 Gradient Boosting Algorithm

(1) GRADIENT BOOSTING ALGORITHM

The generic gradient boosting algorithm is shown in Figure X. It follows a sequence of three distinct steps, with step two being performed for each iteration. At the beginning, an additional initiation of the dataset and a Loss Function is necessary. The mathematical representation of the dataset is like the one used for Decision Tees. A summary: x_i present the explanatory features while y_i represents the corresponding label for one data point of the input dataset N with a total number of n samples.

The mathematical goal of the Algorithm is to reconstruct the unknown functional dependance f between x_i and y_i with an estimate $\hat{f}(x)$ for every data point, such that the specific Loss Function $\Psi(y, \theta)$ is minimized (1) [15, p.1189] [16, p. 2.1].

(1) $F^* = \arg\min_F L(y, F(x))$

The Loss Function is an indicator for the quality of the model. A small Loss for a data point means that the prediction is close or identical to the observed Label and the model therefore categorizes the sample correctly whereas a high Loss implies that the model could not predict the sample well. Given a particular learning task and dataset, different Loss Functions must be considered as Loss Functions are only suitable for specific data and task constellations. The most common Loss Function for Binary Classification is the so-called Bernoulli Loss (2). The Bernoulli Loss can be transformed into a log(odds)-prediction (3) as it is better suited for further calculations. Variations of (3) will be used in the following section to demonstrate the Gradient Boosting Procedure [16, p. 3.1].

(2) BERNOULLI LOSS

(3) BERNOULLI LOG(ODDS)-PREDICTION

Additionally, a Machine Learning Algorithm must be defined as a Weak Learner. For GMB there are multiple Leaners to choose from, again the choice is mostly depending on the prediction task and available data [16, p. 3.2]. A classical approach is the use of Decision Trees, which was also chosen as the Weak Learner for this project. Decision Trees used for Gradient Boosting are always Regression Trees, regardless of whether they are used for

Regression or Classification Problems. The optimization parameters are almost identical as for standalone Decision Trees, but the Trees often look very different because they are specifically created as Weak Learners. As a result, the Decision Trees often only consist out of very few layers with only 8-32 Leaves.

Algorithm Procedure

To showcase the Gradient Boosting Algorithm the same sample dataset is used as for Decision Trees. It again consists out of 8 samples with two features and two categories as target Labels.

DATASET

The first step 1) is to set an initial prediction for all samples of the dataset. The initial prediction is not unique for individual samples but a uniform value. The optimal initial prediction can be calculated using the following equation (4). For $F_0(x)$, representing the initial prediction, a minimum of γ is searched for. The right-hand side of the equation only consists out of a sum for each sample i (of the total dataset N) of the known Loss Function with the respective Label y_i and γ as its input. To find the low point of the equation, the derivate of the Loss Function is required. The final calculation of the overall $\log(odds)$ that as song is classified as "hiphop" is the $\log_e$ of the sum of songs of the category "hiphop" divided by the sum of the songs of the category "jazz" (4). The result can be checked graphically as is equal to the $x1$-coordinate value of the low point of the $\log(odds)$-prediction (5).

(4) $F_0(x) = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$

$F_0(x) = \log_e\left(\frac{5}{3}\right) = 0,51$

(5) COORDINATE SYSTEM

With the completion of step 1), one can start with step 2) of the algorithm. 2) is a sequential process of constructing the Regression Trees with a total of M iterations. The first iteration starts with $m = 1$.

First, the log(odds)-prediction must be converted back into a probability p with the help of a Logistic Function as probability is easier to use for Classification (6). The result is that all songs have the probability of $0,63$ to belong to the category "hiphop".

(6) $p = \frac{e^{\log_e(odds)}}{1 + e^{\log_e(odds)}} = \frac{e^{\log_e\left(\frac{5}{3}\right)}}{1 + e^{\log_e\left(\frac{5}{3}\right)}} = 0,63$

In a) the Pseudo Residuals r_{im} for each sample i of the dataset are created. The equation (7) for calculating the PR consists out of known fragments. For every sample i a PR

r_{im} is calculated using the derivative of the $log(odds)$-prediction with the Label y_i and the Prediction of the last iteration $F = F_{m-1}$ as its input. Again, the equation can be transformed into a very simple equation (8). For each sample the PR can be calculated by only subtracting the previously calculated probability p from the observed Label y. Ideally, an additional column is created in which the PRs are temporarily stored (figure 9).

(7) $r_{im} = -\left(\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right)_{F=F_{m-1}}$

(8) $r_{im} = (y_i - p_i)$

FIGURE PSEUDO RESIDUALS

b) constructs the Regression Trees put of the features of the samples with the corresponding PR as the label. For this example, only Tree Stumps are created to simplify the implementation. The Regression Tree for the first iteration is shown in Figure 10. After the completion of the Tree, Terminal Regions R_{jm} must be defined for every Leaf. j starts with 1 and is increased for every Leaf [15, p.1195].

FIGURE REGRESSON TREE

Following the completion of the Regression Tree, Output Values $\gamma_{j,m}$ are calculated by using the equation presented in (11). For each Leaf in the Tree $\gamma_{j,m}$ is computed by finding $\gamma_{j,m}$ that minimizes the Loss Function (1ß). Like for the initialization step, the derivative has to be created and must set equal to 0. And again, after a complicated transformation, a very simple equation remains (12). The $\gamma_{j,m}$ can be calculated using only the PR and the most recent predicted probabilities p for all samples in the Leaf (11).

(10) $\gamma_{jm} = \arg\min_\gamma \sum_{x_i in R_{ij}} L(y_i, F_{m-1}(x_i + \gamma))$

(11) $\gamma_{jm} = \frac{\sum r_{im}}{\sum p_i*(1-p_i)}$

Part d) marks the end of the first iteration and creates a new prediction $F_m(x)$ for each sample. The new $log(odds)$-prediction is based on the last $log(odds)$-prediction plus the Learning Rate v multiplied by the Output Value(s) for the sample of the last Regression Tree (14) [15, p.1203]. Normally, there is only one Output Value for a sample which makes the summation sign obsolete. The Learning Rate v is a hyperparameter for Gradient Boosting. For this example, a high v is used to better visualize the changes. In practice a v in the order of 0.1 is common as decreasing the Learning Rate tends to give better results [15, p.1206].

(13) $F_m(x) = F_m x - 1 + v * \sum_{j=1}^{J_m} \gamma_{jm} I(x in R_{jm})$

Step 2 is repeated until M is reached. M marks the completion of the training of the Gradient Boosting Model. The $F_M(x)$ is the final prediction for every sample. Lastly, the

prediditions must again be transformed to probabilites like for the calculation of the PR. For the final probabilities thresholds are used to compute the category to which the samples belong [15, p.1204]. A typical threshold for binary Classification is 0.5 as it splits $F_M(x)$ into two equal classes. This process also takes place for unknown samples as described in step 3).

Step 3) is the final step for Gradient Boosting and classifies unknown datasets. An unknown sample gets initialized by $F_0(x)$ and is sequentially routed through every model. For each iteration the Prediction is updated using the Output Value of the Regression Tree. Finally, the last probability is used to classify the sample using the predefined threshold.

2.5.2 Evaluation of Gradient Boosting

Gradient Boosting is a very powerful method as it can effectively capure complex dependencies for various Machine Learning Problems. GBMs enhance already existing Machine Learning Algorithms by solving many problems of single model approaches by relying on a sequence of less complex models. This approach on the other hand comes with its own disadvantages.

The main benefit over Decision Trees is the stability of Gradient Boosting. While large trees always have to make tradeoffs between detail and overcategorization, gradient boosting can gradually get to a deeper and deeper level of detail thanks to small trees with overall better generalization. Furthermore the flexbility of Gradient Boosting and Boosting in general is massive as it only represents the framework with many parameters to adapt the algorithm very specifically to the usecase.

The drawbacks of Gradient Boosting often arise in practice. Gradient Boosting has a significantly higher memory consumption and build time as the model must be constructed sequentially. Also the evalutation is more time consuming as the sample must be processed by each model. From a business perspective Gradient Boosting also has its disadvantages. While the prediciton is better, it is much more complex to evaluate the model and explain the results [16, p. 7.2].

2.6 Cross Industry Standard Process for Data Mining

2.7 Application Programming Interfaces

This section gives an overview over the basic concepts[17, S.1] and technologies behind Application Programming Interfaces (APIs).

2.7.1 Purpose and Usage

An APIs is an interface between two pieces of software.[17, S.1] These might run on the same machine and communicate locally, in the case of a desktop application for example, or on seperate machines that are connected via some network, e.g. in a client/server application.

APIs provide a readily implemented solution to a problem in programming and can be reused , how the API can be used to solve it. APIs such a problem might be finding some value in an array, fetching a file from a hard drive, or getting the latest weather data from a weather service. \@expl@@@filehook@file@pop@assign@@nnnn sections/02_Fundamentals05_api.texsections/0

2.8 Basic Concepts of Music Theory

Since this paper aims to use Music streamed on Spotify to create a machine learning model it is also necessary to briefly discuss the Basics of Music. Therefore, first a definition for music is given and after that the basic characteristics of music are analyzed. Secondly methods are shown that help to classify music.

Music itself is a form of art that combines either vocal or instrumental sounds, sometimes both to express an emotion or convey an idea.[18] Music plays a key factor in many cultures of the world and is unique and different in each of them. But despite being able to define music it can be quite difficult even for artists itself to describe it and they struggle to put their perception into words. Famous Tenor and Saxophonist of the 1950s/1960s John Coltrane described Music as follows. "My music is the spiritual expression of what I am my faith, my knowledge, my being. When you begin to see the possibilities of music, your desire to do something really good for people, to help humanity free itself from its hang ups. I want to speak to their souls."[19]

2.8.1 Melodical Components of Music

Based on John Coltrane's quote, music could colloquially be described as something, "that is used to trigger human sense".[19] To achieve this music consists of various elements that work together harmoniously to form a pleasant-sounding melody. In music theory there are different opinions about the number of elements, which again only illustrates how complicated grasping is from the point of view of analysis. Many factors are highly interdependent and correlate with each other.

- **Pitch and Melody**: Pitch is the word used to describe the highness or lowness of a musical sound. A series of Pitches, also described as scales, is used to create a Melody. Melodies can be derived from various scales for e.g., the traditional major and minor scales of music or even more unusual ones like the whole tone scale.[20, p. 86] Furthermore melodies can be described in two ways. Conjunct melodies are smooth and easy to sing or play. Disjunct melodies however represent the opposite. They are ragged or jumpy and difficult to sing or play.

- **Harmony and Chords**: Harmony can be described as the sound created when two or more Pitches are performed at the same time to form a chord. Chord is simply the definition for three or more notes sounding at once. Chords themselves are used to create a musical mood. Harmony can be split up into two categories taking the sound of pitches into consideration. Harmony can sound Consonant, which means the pitches sound pleasant together or Dissonant, meaning the pitches sound unpleasant together.[20, p. 94]

- **Rhythm**: Rhythm refers to the recurrence of notes and silences in time. A rhythmic pattern is formed by a series of notes and silence repeats. In addition to signify when notes are played, rhythm also defines how long notes are played and with what intensity. The difference in time creates varying note durations as well as different type of accents.[21]

- **Texture**: Texture indicates the number of instruments or voices that are used to contribute to the density of the music. Texture can be split up into 3 types of Monophonies, Homophony and Polyphony. Monophony describes a single layer of sound e.g. a solo voice. Homophony is a melody with an accompaniment which can be a lead singer with a band or a solo singer an a guitar or piano. Polyphony is a form of texture which consists of two or more independent voices. One voice forms the melody and the other forms a support role. This could be for example a lead singer with a choir.[22]

- **Timbre**: Timbre, also known as tone colour or tone quality, can be described as the specific tone or quality an instrument or a voice has. Timbre helps to distinguish instruments from each other when playing the same melody or simple notes. For example, a "C" Note on the Piano and a sung "C" have the same pitch but different sound quality which gets differentiated by timbre. Timbre usually can be described with adjectives used to describe color, temperature, or the human voice e.g., warm, cold, metallic, harsh, dry.[20, p. 94]

2.8.2 Lyrics and Instruments

Another way to characterize music is Text. Lyrics are the definition for the linguistic part of a song. Lyrics usually consist of verses and choruses and can be implicit or explicit. More or less every song uses unique lyrics but nevertheless we can categorize lyrics by topics they address.[23] Statistics have shown that the most common themes are Growing Up, Friendship, Statements of Discontent, heartbreak or Death.[24] Another factor that must be taken into consideration when characterizing music are instruments. In the previous chapter we already talked about the fact that timbre is used to distinguish instruments from each other. The timbre focuses mainly on the sound of the instruments to distinguish them. However, we can distinguish instruments not only by the way they sound but also by how the sound is produced in the first place. This is done with a classification of the instruments into 5 categories.

- Idiophones: Sound gets produced by the body of the instrument vibrating e.g., xylophones

- Membranophones: Sound produced by vibration of a tightly stretched membrane such as drums

- Chordophones: Sound produced by vibrating strings e.g., piano or cello

- Aerophones: Produce sound by vibrating columns of air such as the pipe organ or oboe

- Electrophones: Produce sound by electronic means e.g., synthesizers or electronic instruments.

Independently of the tone production, a classification according to the type of player is also possible, i.e., wind instruments, percussion instruments, string instruments, keyboard instruments, plucked instruments.[25]

2.8.3 Chronological Classification

Not only compositional aspects can be used to distinguish music, but also the history of music provides a basis for this. This is done with the help of epochs. In music, an epoch is defined as a period in which stylistic similarities prevailed. However, epoch terms are a bit problematic because they give the impression that different styles were abruptly replaced. This is not the case, because with different styles, which often do not resemble each other at all or even contradict each other, there was often a smooth transition or simultaneous

coexistence. Moreover, epochs are usually only generic terms for many undercurrents. The epochs that are considered relevant and for the most part include all styles since the 8th century are the Middle Ages, Renaissance, Baroque, Classicism, Romanticism, and the modern era.[26] Although the music classifications of the classical and modern eras occur together in history, there are several differences in distinguishing music. While classical eras could often be delineated by, for example, the preferred use of instruments, the increased emergence and sheer diversity in music since the 20th century has made it almost impossible to bundle all styles under a single epochal term. For this reason, genres were developed for the classification of so-called "new music".[27]

2.8.4 Genres

Musical genres assign different pieces of music with common characteristics to a unified tradition, history, or convention. Thus, a genre combines songs that sound the same or emit the same feelings, as well as songs that are based on the same characteristics of lyrics and instruments in songs. Dividing music into genres has become standard in today's music industry and due to the diverse number of possibilities in the creation of songs, there are now already over 1800 recognized specific genres with various subcategories. These subcategories or also called subgenres have again basic characteristics of the main genre as a basis but also their own characteristics to clearly distinguish themselves in their own genre and are often also often called style of the genre. These subgenres in turn have subcategories so that somewhat of a "genre tree" can be formed. The sheer number of genres and subgenres reflects the diversity of music. In this thesis the focus will be on the 3 main genres Hip Hop, Jazz and Rock which will be discussed in more detail below.[27]

Hip Hop

Although Hip Hop is often widely considered as a synonym for rap music it can rather be defined as somewhat of a cultural movement or a form of lifestyle that includes a music genre. Hip Hop culture consists of various elements which are united under this umbrella term. Elements are djing/turntablism, rapping, beatboxing, breakdancing, and graffiti/visual art.[28] These typical street style elements or activities created a cultural revolution and shaped music styles, fashion, technology, art, and more.[29] Even to this day hip hop continues to develop new art forms that impact the lives of new and old generations. When it comes to the history of Hip Hop the culture has its origins in the 1970s where it first appeared in African American ghettos. Because of this heritage, hip hop still sees itself as street culture, and musicians who practice this form of music often have close ties to gangs, clans, and poverty.[30] Basic elements of hip hop come from the genres

of funk and soul, which are also influenced by African Americans. Especially because of the distinctive rhythms, these genres offered themselves very well as a basis. Through the development of hip hop, however, it is also becoming more and more unique in type and form. The main characteristic of the music of hip hop is the interaction between a rapper and a beat. Rarely are real instruments used for the music and beats are usually created by electronic instruments or samples which is, the use of already finished or existing sound/music recordings. Beats can be very diverse and can represent many moods, e.g., aggressive, relaxed, harsh, etc.[28] But the main focus in rap is usually on the rapper himself and even more on the lyrics. Rappers use rhythm, lyrics, and the timbre of their voice to express themselves. The Artists use their voice as an instrument and more importantly different voice pitches to adapt to the intention of the lyrics. Rappers are often measured by the scene on their so-called "flow", the ability how quickly and how smoothly they can perform chants without errors. The themes of Hip Hop songs are often poverty, drugs, violence, struggles, or righteousness. Since many hip-hoppers come from ethnic and racial minorities, they often use personal life experiences for their lyrics. In addition, many songs also want to convey a message that often revolves around necessary political changes, social justice, or grievances in society.[31]

Jazz

Jazz is a musical style in which improvisation plays a central role. In many jazz performances, artists often play pieces they just made up from their heads and perform them on spot. Jazz has its historical roots in the early 19th century America more specifically in New Orleans. The city was an ideal breeding ground for jazz music because of given cultural diversity. The percentage of ethnic minorities was much higher in this city than elsewhere in America, which is why it was often called a melting pot of cultures. African Americans, people from the Caribbean, European immigrants, and sometimes even white Americans usually lived in the same neighborhoods and racially segregated ghettos that often existed in other American cities were not present here.[32] However, the biggest influence on Jazz had the Afro-American culture as the music somewhat reflected the breaking away from previous rules of slavery and oppression. The improvisation of songs should act symbolically as a counter to the rules they had to deal with for a long time. Since its inception, jazz has gone through many different phases. The beginnings of jazz, from the 1910s to the 1920s, were characterized by small bands, often consisting of only a frontman and a few accompanists. The frontman often improvised pieces using a cornet or a trumpet, while the accompanists supported him with clarinets or trombones. In addition, often instruments like the banjo, piano, double bass, or drums were used to create a rhythm. In the 1930s and 1940s, the Swing and Big Band era emerged. Now, for the first time, singers appeared before big bands and bandleaders, and the clarinet was largely replaced by the

saxophone. Moreover, the jazz epicenter shifted from New Orleans further and further to New York.[33] The 1950s and 1960s then introduced laid-back cool jazz in contrast to the more fast-paced songs of the previous decades. Here a jazz quartet often played soothing and slow songs. With the introduction of electronic instruments in jazz from 1970 onwards, many subgenres were formed, such as jazz-rock. Until today, there are many forms of jazz, which are mainly oriented to the New Orleans origins and the 3 mentioned eras. Jazz has its main musical roots in the blues, but there are also elements of rock and classical music in it.[34] A distinctive rhythm is a key characteristic of jazz music, these are created mainly by "swinging" eighth notes. The so-called swinging is created by emphasizing one note of the eighth note pair while the second note is lighter and "swings" to the next note. Jazz is also very polyphonic, which means that many sounds are played simultaneously and as a result, various layers of harmony are laid over an initial basic melody. In Addition to the use of classical European instruments such as the saxophone or trumpet, instruments such as drums, bass, keyboard, guitar, and trombone are often used. Some types of jazz also have front singers, but often pieces are played without vocals.[35] As already mentioned before. As already mentioned before the main characteristic of Jazz is the spirit of improvisation. This unifies almost all forms of jazz music. All members of a jazz band can be asked to improvise on a jazz tune when performing it. In addition, jazz artists attach great importance to imprinting their own sound and style on the music, so they usually even play their own songs slightly modified or with a distinct style. This leads to the fact that thousands of jazz recordings can be found for the same song, but they all sound different. Finally, it is important to mention that jazz can also reflect many different emotions. Everything from pain to joy is possible. For many People of Colour, it resembles the feeling of freedom, because for them, as mentioned above, jazz represents a strong voice against suffering, oppression, and injustice.[36]

Rock

The musical Genre Rock is a popular music genre that combines elements of rhythm and blues, jazz and country music while adding electric instruments. It originated as Rock 'n Roll in the late 1940's and early 1950's and of course also is constantly changing and evolving. The basis of rock formed the music genres blues, gospel and country. Early rock 'n' roll came from cities like Memphis, Chicago, New Orleans or St. Louis. However, the genre spread very quickly throughout Western culture, and so it was mainly the British who liked the genre very much and who, in turn, took it to new heights.[37] British bands like the Beatles or the Rolling Stones emerged and became very popular in the USA as well. Rock dominated the music with its loud, dynamic, energetic and intense style for almost over 50 years until it was replaced by hip hop as the most mainstream genre. One of the main characteristics of rock is the infectious beat and rhythm. The music was primarily designed

for dancing and was a clear distinction from other music genres popular at the time, such as jazz or swing. Thats why from its start in the 50s the genre was very popular among young people as they felt they could break out of rigid traditions. Furthermore it is also important to mention that rock, like hip-hop later on, not only had an impact on music but also a cultural influence on the generations from the 1950s to the 1990s.[37] Rock especially supported the sense of rebellion and social justice in the western world of the 1960s and influenced clothing style, hair style and even attitude of its listeners for over 40 years. The older, more conservative generation at the time, which favored more quiet songs, rather detested rock. The energy already mentioned above is a unique characteristic that sets rock apart from many other genres. Rock music is often very wild, impulsive and driving. But what defines rock the most is the use of electric instruments and especially the use of the electric guitar.[38] The indispensable electric guitar makes rock probably the genre that is most influenced by a single instrument. Pioneers of rock like Elvis Presley, Jimi Hendrix or Chuck Berry experimented a lot with the electric guitar and used the unique sound to their advantage. Hendrix in particular often used the so-called "scream" of the electric guitar as his trademark, where he plucked the string in such a way that a shrill sound could be heard through the amplifier. This instrument allowed the artists to reach new melodic aspects and pitches that are not achievable with the use of pure acoustic instruments.[38] Rock music is typically performed only in bands with a lead singer. He usually plays an electric guitar himself and is accompanied either by other electric guitars, normal guitars, other electric instruments and a drum kit. Lyrical texts of rock are also very diverse due to its division into many subgenres. Lyrics may not be very profound other lyrics by artists like Bob Dylan, however, are considered comparable to fine poetry. Rock music subgenres are very different and vary greatly in terms of rhythm and tempo. For example, the music genre of heavy metal is almost incomparable with soft rock.[39]

3 Implementation

3.1 Data Collection

In this section the approach and implementation of data collection for this project is examined.

3.1.1 Requirements for the dataset

Basic requirements the dataset should fulfill are

- **Includes Spotify song features**

  Spotify provides a set of song features that were generated using their own models. The dataset should inlude these features, as they are needed to train the model

- **Includes genre as label**

  The dataset needs to include the genre of the track to use as a label for the classifier

- **Has sufficient sample size per genre**

  In order to train the model well, a sufficient sample size is needed per genre. It was not known before collecting the data, how many samples were enough.

- **Song and Artist name**

  The best way to filter out duplicates is to use the song and artist names. Spotify does provide a track id for each song, however, if a song is released twice (e.g. as a single and later in an album), these track ids will differ which will lead to a duplicate entry.

Additional fields are not going to be used in this analysis, but might still be collected in order to publish the dataset and enable others to use it for different applications.

3.1.2 Existing Datasets

As this paper examines creating a model specifically on Spotify Song Data, a search on the internet was conducted first, to find a potential pre-made dataset, pulled from the Spotify API, which could be used. Kaggle [1] lists an extensive catalogue of community provided datasets, so the main sources of this search were Kaggle and Google search for the term "Spotify Song Data". Kaggle lists a couple of datasets that could be applicable to the research question in this paper. Some examples of datasets listed are given and explained, why they could not be used for this project.

"Spotify music analysis" by user Aeryan [2] is a dataset of 2017 rows, which includes musical features like acousticness and tempo, the song title and artist, but lacks a genre field. Because of the small sample size and the missing genre field, this dataset could not be used.

There are multiple datasets which include songs that were featured in Spotify's "Top 50" Playlists, charts, or year in review, recorded at a single point in time or historically. [3] [4] These could not be used, as the sample size is again too small and the focus is specifically on the most popular tracks and not a wide variety of music in a genre.

"Dataset of songs in Spotify" [5] is the most promising dataset examined, as it has a big sample size and includes genre data. However, the methodology of how the data was collected is not included and there could be multiple ways of how genre data for a given song is collected, as is explained later. Also the genres are limited to very specific directions of Electronic Dance Music and Hip-Hop.

As no optimal dataset for this research paper could be found using our search criteria, a dataset was specifically created for this paper using the Spotify Web API.

3.1.3 Ressources and Approach

Spotify provides extensive documentation for developers on their developer website [6]. This includes development and design guidelines for teams, that want to integrate Spotify's service into their own apps, documentation on IOS and Android development a community forum, a developer dashboard and the Web API documentation, which is the main ressource for data collection from Spotify.

---

[1]   Kaggle Website: https://www.kaggle.com/
[2]   https://www.kaggle.com/aeryan/spotify-music-analysis
[3]   https://www.kaggle.com/nadintamer/top-spotify-tracks-of-2018
[4]   https://www.kaggle.com/leonardopena/top50spotify2019
[5]   https://www.kaggle.com/mrmorj/dataset-of-songs-in-spotify
[6]   https://developer.spotify.com/

The API is based on the Representational State Transfer (REST) architecture. The different endpoints return JSON metadata directly from the Spotify Data Catalogue [40]. There are also features to query for user related data using an authorization flow with the users Spotify account, but this is not relevant in this context. [40] Requests to the API are made via HTTPS GET or POST methods. The API can be used by anyone, but authorization via the OAuth protocol is required to access data from the API. To explore the API and find endpoints to use, Spotify provides a developer console, which can be used to send requests and see what kind of responses come back. This is not suitable for saving the data or making multiple requests programmatically, but is helpful for API exploration. As there is not one single endpoint that delivers all required fields, multiple queries that build on top of each other have to be made.

The approach began with using the API reference to get an overview over the endpoints and their responses. The specific endpoints that might return interesting data were queried using the Spotify Web Console, to see a response with live data and which exact fields are returned. Beyond the Web Console, the tool "Postman" was used to explore the API. It is a platform that can be used to make HTTP requests to an API and store these requests in a collaborative environment. [41] It supports the required authorization workflows and enabled the research team to explore endpoints together, easily make API calls without having to authenticate by hand, and save the endpoints and required input parameters in a shared workspace. Once exploration was complete, the complete data collection was implemented in Python 3, mainly using the libraries http, json and requests. The final result was saved as a CSV file to be used for data exploration and further processing.

Many API endpoints give the option to specify a country and locale that the results should apply to. Whenever possible these values were explicitly specified to eliminate the risk of receiving data for multiple different countries or languages, which would have made the dataset incoherent. For country, we used the country tag "US" which means that the API replies only with datapoints that are applicable to users in the United States of America. For locale we used en_US in order to receive the data in the english language. These two values differ in that country changes the actual entries from the database that are selected for the response and sent to the users. Locale defines in which language these selected entries are returned in.

3.1.4 Authorization

Using the Spotify documentation for authorization workflows [42], authorization was first tested using Postman and then implemented in Python. The workflow is explained using

the Python code. Using a Spotify account to log in, the developer dashboard can be accessed. Here an application was registered with Spotify for the research project. Spotify tracks API usage per application and can recognize if the API is being abused or too many requests are sent, which will result in rate limitation or blocking. On the API Dashboard, a "Client ID" and "Client Secret" can be retrieved. These credentials are used to start the authorization flow, as described by Spotify in Figure 1.

Figure 1: Spotify Authorization Flow
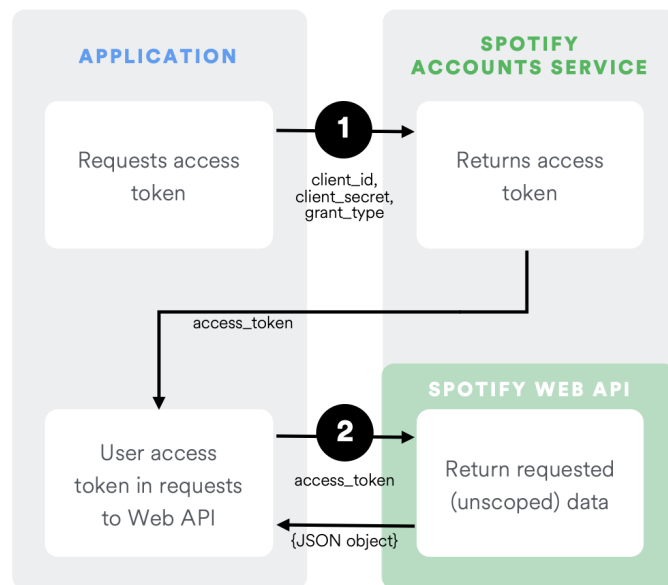
Source: [42]

Step one is a post request to the Spotify Account Service with the client id and client secret from the application dashboard, which returns an access token, that is valid for one hour. This token can be used to access any endpoint of the actual API that does not require user specific data. When the token expires, a new one has to be requested before querying the API again. Figure 2 shows the full request and response to acquire the token.

Figure 2: Access Token Request

POST	**https://accounts.spotify.com/api/token**
	request access token

Body	application/x-www-form-urlencoded

```
1  {
2      "grant_type": "client_credentials",
3      "client_id": client id from application dashboard,
4      "client_secret": client secret from dashboard
5  }
```

Response	application/json
200 ok	

```
1  {
2      "access_token": "BQDgQCSx-tIMDo9LfVeZxm6YMl2p_WbEU3Q
           9ENsVl7e--6d_vockTsfzMVUhPWiHSSnFUuHvm_9POA1kYEw"
           ,
3      "token_type": "Bearer",
4      "expires_in": 3600
5  }
```

3.1.5 Getting Features

In order to predict the genre of a track based on audio features, these features have to be requested for every track. Spotify provides an endpoint to get audio features for a single track or up to 99 tracks at a time. The latter is used in the Python implementation as it reduces the number of requests to be made. The typical request/response pattern for the audio-feature request endpoint is shown in figure 3.

Figure 3: Audio Feature Request

GET	**https://api.spotify.com/v1/audio-features?ids={ids}**
	request audio features for id

Parameter

ids	comma seperated list of up to 99 song ids

Response application/json

200 ok

```
1  {
2      "audio_features": [
3          {
4              "danceability": 0.677,
5              "energy": 0.638,
6              "key": 8,
7              "loudness": -8.631,
8              "mode": 1,
9              "speechiness": 0.333,
10             "acousticness": 0.589,
11             "instrumentalness": 0,
12             "liveness": 0.193,
13             "valence": 0.435,
14             "tempo": 82.810,
15             "type": "audio_features",
16             "id": "2e3Ea0o24lReQFR4FA7yXH",
17             "uri": "spotify:track:2e3Ea0o24lReQFR4FA7yXH
                   ",
18             "track_href": "https://api.spotify.com/v1/
                   tracks/2e3Ea0o24lReQFR4FA7yXH",
19             "analysis_url": "https://api.spotify.com/v1/
                   audio-analysis/2e3Ea0o24lReQFR4FA7yXH",
20             "duration_ms": 211497,
21             "time_signature": 4
22         },
23         ...
24     ]
25 }
```

With the exception of type, id, uri, track_href and analysis_url, all of the fields included in this response can be used as features in the dataset. However, this api call expects a track id, which we need to get using other api calls first. This could be a search endpoint, getting all tracks in a playlist, etc. Also, it does not give the track or artist names and doesn't include a genre.

3.1.6 Getting Track IDs and Labels

There is no simple endpoint that takes one or more track ids and returns a "genre" field in its response. The exploration of the API using the reference, web console and Postman only revealed two ways of getting the genre of a track.

The first way is using the artist of a track. Given a track id, the artists of the track and their corresponding ids can be requested by using the "/tracks/id" endpoint. Then, using the artist id, the genres that an artist is known for are returned, as can be seen in figure 4.

Figure 4: Artist Request

GET	**https://api.spotify.com/v1/artists/{id}**
	request information about an artist by their id

Parameter

id   id of the artist

Response   application/json

200   ok

```
1  {
2      "external_urls": {
3          "spotify": "https://open.spotify.com/artist/6l3
               HvQ5sa6mXTsMTB19rO5"
4      },
5      "followers": {
6          "href": null,
7          "total": 15554811
8      },
9      "genres": [
10         "conscious hip hop",
11         "hip hop",
12         "north carolina hip hop",
13         "rap"
14     ],
15     "href": "https://api.spotify.com/v1/artists/6l3HvQ5
               sa6mXTsMTB19rO5",
16     "id": "6l3HvQ5sa6mXTsMTB19rO5",
17     "images": [
18         ...
19     ],
20     "name": "J. Cole",
21     "popularity": 89,
22     "type": "artist",
23     "uri": "spotify:artist:6l3HvQ5sa6mXTsMTB19rO5"
24     }
```

This examplary API response shows a problem with this approach. One artist can be sorted into multiple genres. A given track might be associated with either of the artists genres, but the data does not show, which one exactly. Additionally a track might have multiple artists which further complicates this. Given these circumstances, this approach is problematic.

The second way is Spotifys "categories" feature. The apps search tab provides a number of categories that a user can browse through to find new music in their preferred genre or style. In figure 5a an overview over some of the categories that are available in the Spotify App is shown. There are categories of multiple types, e.g. specific activities, like Cooking or Gaming, or places, like "At Home" or "In the Car". But there are also categories for nearly all major genres. In the app screenshot there is for example Classical, Jazz or Soul. These categories can be used to get tracks that belong in each specific category. When a user taps on one of the categories, playlists that contain tracks of the respective category are shown to the user, like shown in figure 5b.

Figure 5: Categories and Playlists in Spotify App

(a) Category Overview   **(b)** Playlists in category
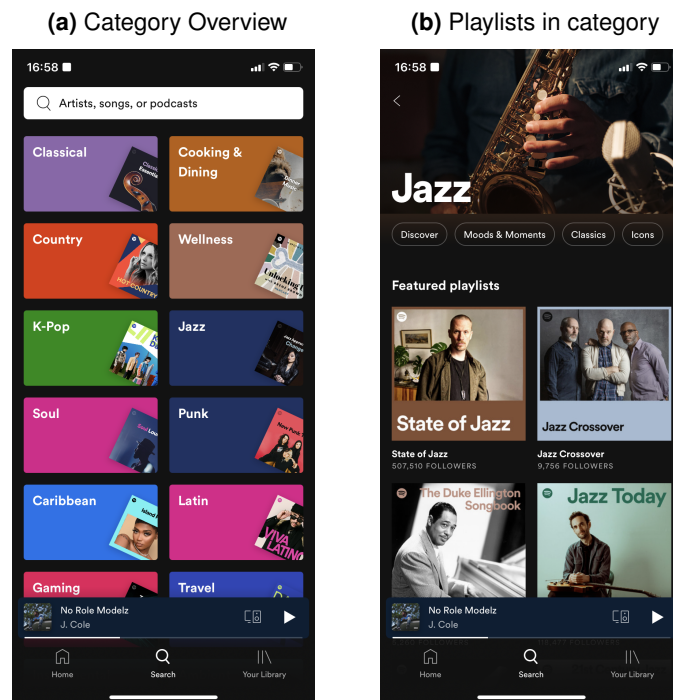
The API mirrors the apps behaviour and provides an endpoint to get a list of categories and their ids, one to get all playlists and playlist_ids in a category, and one to get all tracks and track_ids in a playlist. This chain of API calls is used to request every track in every playlist in a certain category.

3.1.7 Python Implementation

This section describes, how the previously explained API-Calls are used to request data from the API and save the complete dataset as a CSV file. The script was implemented in

a way that lets anyone with a registered Spotify developer application run it to gather their own data. It supports filtering for certain genres to speed up the process and exporting the current data after each step to be able to pause the data collection and check the current results or continue later. The details of these functions are explained in this section using code snippets.

The Python requests library is used to execute the requests, as it supports the features needed to send the GET and POST requests we need without having to write complicated code. The dotenv library is used to read the client id and secret from a seperate .env file, rather than writing it into the code. This prevents these sensitive credentials from being committed into version control, which is hosted in a public GitHub repository and would therefore make the credentials public. The "json" library is used for transforming Python dictionaries into json data, "math" for access to rounding functions, "os" for managing system paths and access to files, "http" to define retry and timeout behaviour of the API-Calls and "csv" to finally transform the json data into a CSV file.

First, the credentials are read using "load_dotenv". Then a method is defined which takes the client id and secret as input, executes a POST request to the accounts API, as shown in 2 and return the access token from the "access_token" field in the response.

```python
#Get environment variables from ".env" file and read credentials
load_dotenv('.env')
client_id = os.environ.get('CLIENT_ID')
client_secret = os.environ.get('CLIENT_SECRET')

# Authenticate and get an API Token from Spotify using a Client ID and secret
def getAuthTokenFromCredentials(id, secret):

    url = "https://accounts.spotify.com/api/token"

    payload = f'grant_type=client_credentials&client_id={id}&client_secret={secret}'
    headers = {
    'Content-Type': 'application/x-www-form-urlencoded',
    }

    response = requests.request("POST", url, headers=headers, data=payload)

    return response.json()["access_token"]

auth_token = getAuthTokenFromCredentials(client_id, client_secret)
```

Next the http library is configured to use a ten second timeout and attempt five retries. If a request to the API is made and no response is given from Spotify's servers after ten seconds, the request will be sent again. If there is still no answer from the server after five attempts, an Exception is thrown and the script ends. Timeouts are crucial as short

disconnections from the Internet could result in the whole data collection step failing. The http setup code is found in appendix **??**.

The next part of the script implements the filter mechanism

```python
category_filter = None

# comment this out if you don't want to set a filter
category_filter = ["hiphop", "pop", "country", "rock", "latin", "rnb", "mood", "
    indie_alt",
                   "regional_mexican", "edm_dance", "inspirational", "chill", "
                       party", "roots",
                   "kpop", "instrumental", "ambient", "alternative", "classical",
                       "jazz", "soul",
                   "punk", "blues", "arab", "afro", "metal", "caribbean", "funk"]

# tells data collection functions to not use filter if it's not set
if category_filter is not None:
    use_filter = True
else:
    use_filter = False
```

The option to filter the categories that are to be taken into account when collecting the data is given. Possible values are all valid category ids, which can be requested using the API call shown in figure 6

Figure 6: Categories Request

GET	https://api.spotify.com/v1/categories?country={country}&locale={locale}&limit={li
	Get a list of categories used on Spotifys browse tab

Parameter

id	id of the artist
country	only display categories available in a specific country
locale	the language in which the categories should be returned
limit	the maximum number of items to be returned
offset	index of the first item to return

Response application/json

200 ok

```json
 1  {
 2      "categories": {
 3          "href": "...",
 4          "items": [
 5              {
 6                  "href": "...",
 7                  "icons": [
 8                      {
 9                          "height": 274,
10                          "url": "...",
11                          "width": 274
12                      }
13                  ],
14                  "id": "hiphop",
15                  "name": "Hip-Hop"
16              },
17              {
18                  "href": "...",
19                  "icons": ...
20                  "id": "pop",
21                  "name": "Pop"
22              },
23              ...
24          ],
25          "limit": 10,
26          "next": "...",
27          "offset": 0,
28          "previous": "...",
29          "total": 58
30      }
31  }
```

The response shows an id field for each item in the items array. These ids can be used in the filter list. If the category_filter variable is undefined, no filter is set.

Next, the option to import existing data from a file is given. If a user already has the json data containing all playlists for all categories they want to use, they can load the json file here and skip executing the collection of categories and playlists and continue right at the collection of tracks for each playlist.

```
1    data = {}
2
3    #To load data_object from file instead of rerunning the scripts, uncomment this:
4
5    file = open(os.path.join("data_collection", "json", "tracks_full.json"))
6    data = json.load(file)
```

Next, the four main steps of data collection will be defined and executed:

1. Getting categories

2. Getting playlists

3. Getting tracks

4. Getting audio features

Each step is defined as a function with some input parameters and the resulting json data as output, which can then be used as input for the next step.

The function definition for getting the categories looks like this:

```
1    def getAllCategories(
2        requests_session, # takes the requests session, which was set up in the
              beginning
3        auth_token, # this is the token which was retrieved in the authorization step
4        use_category_filter=False, # set to True if the category_filter should be used
5        category_filter=None,  # this takes the list which is used as a category
              filter
6        write_to_file=False, # set to True if the result of this step should be
              written into a json file
7        path_to_file=''): # give the file path at which the json file should be stored
8
9        ...
```

The function first makes a simple API call with limit=1 to get the total amount of categories available. This is necessary, because the API returns a maximum number of 50 entries per request, so if the total number of items exceeds 50, multiple calls have to be made.

```
1    # Establishing the requests session
2    http = requests_session
3
4    # First API call used to get the total amount of categories
5    headers = { 'Authorization': f'Bearer {auth_token}' }
6    url = "https://api.spotify.com/v1/browse/categories?country=US&locale=en_US&limit
         =1"
7
8    try:
9        response = http.request("GET", url, headers=headers, data={})
10       if response.status_code != requests.codes.ok:
11           raise Exception
12   except Exception as e:
13       raise SystemExit(e)
```

The requests_session is saved in the http variable. Then the auth_token is wrapped in an Authorization header conforming to the OAuth Bearer Token standard, which is used by the API. The url is given with the right country and locale settings and the request is sent inside of a try block. If the response doesn't contain an HTTP status code 200, an Exception is raised and the program is stopped, as there is possibly an error in the request or response which could mean corrupted data.

Next, the total amount of categories is extracted from the response and the number of pages is calculated.

```
1    categoryAmount = response.json()["categories"]["total"]
2
3    # API only returns 50 items at a time. Offset can be used to gradually get all
         items
4    # Calculate number of pages with 50 items
5    pages = int(math.ceil(categoryAmount/50))
6    data = {"categories": []}
```

By dividing the category amount by 50 and rounding up to the next integer, the number of API calls necessary to get all entries is calculated. We call this pages, like pages in a book. Page one contains items 0 to 49 and is retrieved by using offset 0 and limit 50. Page two contains items 50 to 99 and is retrieved by using offset 50 and limit 50. This is done as many times as required. the last page will not necessarily contain 50 items but less, as those are left over.

A for loop is used to execute the request for every page as shown before. The offset is increased by 50 with each run. Each category's id and name are read from the response and appended to the data object:

```
1    for x in range(pages):
2        url = f"https://api.spotify.com/v1/browse/categories?country=US&locale=en_US&
             limit=50&offset={x * 50}"
3
```

```
4        try:
5            response = http.request("GET", url, headers=headers, data={})
6            if response.status_code != requests.codes.ok:
7                raise Exception
8        except Exception as e:
9            raise SystemExit(e)
10
11        # categories are stored in the data dictionary
12        for el in response.json()["categories"]["items"]:
13            if (use_category_filter == True and el["id"] in category_filter) or
                use_category_filter == False:
14            data["categories"].append({
15                "id": el["id"],
16                "name": el["name"]
17            })
```

Outside of the for loop, the resulting data is written to a file if write_to_file was set to True and the data object is returned:

```
1    if write_to_file == True:
2        with open(path_to_file, 'w') as outfile:
3            json.dump(data, outfile, indent=2)
4
5    return data
```

The data collected after this step looks like this:

```
1  {
2      "categories": [
3          {
4              "id": "hiphop",
5              "name": "Hip-Hop"
6          },
7          {
8              "id": "pop",
9              "name": "Pop"
10         },
11         ...
12     ]
13 }
```

This concludes the function getAllCategories. The data retrieved from this step is used for the next step, getting the playlists. The function definition here is slightly different:

```
1    def getPlaylistsForCategories(
2        requests_session,
3        auth_token,
4        data_object,
5        write_to_file=False,
6        path_to_file=''):
```

Instead of the filtering options, which only apply to the categories, this function takes the data_object from the previous step. This could be passed in directly from the previous

function or read from a file. The function implements a nested for loop. The outer loop iterates over all categories in the data object. The inner loop uses an endpoint to get each category's playlists. The API request/response is shown in figure **??**.

Figure 7: Get Category's Playlists Request and Response

GET	https://api.spotify.com/v1/categories/{category_id}/playlists?country={country}&li
	Get Category's Playlists

Parameter

category_id	category id
country	only display categories available in a specific country
limit	the maximum number of items to be returned
offset	index of the first item to return

Response application/json

200 ok

```
1   {
2       "playlists": {
3           "href": "...",
4           "items": [
5               {
6                   "collaborative": false,
7                   "description": "New music from YoungBoy
                        Never Broke Again, DaBaby and 2
                        Chainz. ",
8                   "external_urls": ...
9                   "href": "...",
10                  "id": "37i9dQZF1DX0XUsuxWHRQd",
11                  "images": ...
12                  "name": "RapCaviar",
13                  "owner": ...
14                  "primary_color": null,
15                  "public": null,
16                  "snapshot_id": "...",
17                  "tracks": {
18                      "href": "https://api.spotify.com/v1/
                            playlists/37i9dQZF1DX0XUsuxWHRQd/
                            tracks",
19                      "total": 50
20                  },
21                  "type": "playlist",
22                  "uri": "spotify:playlist:37i9dQZF1DX0
                        XUsuxWHRQd"
23              },
24              ...
25          ],
26          "limit": 50,
27          "next": "...",
28          "offset": 0,
29          "previous": null,
30          "total": 50
31      }
32  }
```

As there is an upper limit of 50 playlists per request, the same paging method is used as in the category function. It is checked, if each items "type" field contains the value "playlist", to filter out any items that don't fit the schema. Then each playlists name and id is stored in the data object, which is returned from the function. The json data after this step looks like this:

```
1  {
2      "categories": [
3          {
4          "id": "hiphop",
5          "name": "Hip-Hop",
6          "playlists": [
7              {
8              "id": "37i9dQZF1DX0XUsuxWHRQd",
9              "name": "RapCaviar"
10             },
11             {
12             "id": "37i9dQZF1DX6GwdWRQMQpq",
13             "name": "Feelin' Myself"
14             },
15             ...
16         },
17         ...
18     }
19 }
```

The third step gets all tracks in each playlist. The function definition is the same as before, as this function also takes data from the previous step. This time, there are three nested for loops. One for looping through the categories, then another one for looping through the playlists. In this one there is again calculated how many pages of 50 tracks need to be requested and the third loop requests the pages and appends the tracks to the playlist data. The request parameters for this request differ from the other steps. Instead of country, there is a market parameter, which essentially does the same thing. There is also a fields parameter, which takes a string that can be used to define which fields the API should return. This is done to save bandwith, as the response of this endpoint is very large and most of the datapoints might not be needed by the client. In this case, the value "items(track(name, id, album(name, id), artists)), total" is used to return only items of type track, their id and name, their albums name and id, and their artist information. Also the total amount of tracks in the playlist is returned. There is also an "additional_types" parameter, which can be used to specify if only music tracks, or also podcast episodes should be returned. In this case, only tracks are returned as podcasts are irrelevant to our analysis. An examplary API request/response using the described "fields" and "additional_types" parameters is shown in figure 8. As a track can have multiple artists, each artist's name and id are extracted and saved with the track. As this function is very similar to the previous one, the code is not shown here. It can be found in the appendix.

Figure 8: Get Playlist's Tracks

GET	**https://api.spotify.com/v1/playlists/{playlist_id}/tracks?market={market}&fields={fi**
	Get Playlist's Tracks

Parameter

playlist_id	playlist id
market	only display items available in this market
fields	a string describing which fields to return
additional_types	supported item types. valid types are "track" and "episode"
limit	the maximum number of items to be returned
offset	index of the first item to return

Response application/json

200 ok

```
1  {
2      "items": [
3          {
4              "track": {
5                  "album": {
6                      "id": "4oxmme6i4mypSt2DDzPTsW",
7                      "name": "DS4EVER"
8                  },
9                  "artists": [
10                      {
11                          "external_urls": {
12                              "spotify": "..."
13                          },
14                          "href": "...",
15                          "id": "2hlmm7s2ICUX0LVIhVFlZQ",
16                          "name": "Gunna",
17                          "type": "artist",
18                          "uri": "spotify:artist:2hlmm7s2
                                ICUX0LVIhVFlZQ"
19                      },
20                      ...
21                  ]
22              }
23          },
24          ...
25      ],
26      "total": 50
27  }
```

The data collected after this step looks like this:

```
1  {
2    "categories": [
3      {
4        "id": "hiphop",
5        "name": "Hip-Hop",
6        "playlists": [
7          {
8            "id": "37i9dQZF1DX0XUsuxWHRQd",
9            "name": "RapCaviar",
10           "tracks": [
11             {
12               "id": "2AaJeBEq3WLcfFW1y8svDf",
13               "name": "By Your Side",
14               "album": {
15                 "id": "2RrZgDND03MLu6pRJdTkz5",
16                 "name": "By Your Side"
17               },
18               "artists": [
19                 {
20                   "id": "45TgXXqMDdF8BkjA83OM7z",
21                   "name": "Rod Wave"
22                 }
23               ]
24             },
25             ...
26           ]
27         },
28         ...
29       ],
30       ...
31     },
32     ...
33   ]
34 }
```

The fourth step is getting the audio features for each track. The request that is used was already shown in 3 The function used has the same definition as before but implements one more for loop to iterate through the tracks. Again, the code for this step is found in the appendix.

The data collected after this step looks like this:

```
1  {
2    "categories": [
3      {
4        "id": "hiphop",
5        "name": "Hip-Hop",
6        "playlists": [
7          {
8            "id": "37i9dQZF1DX0XUsuxWHRQd",
9            "name": "RapCaviar",
10           "tracks": [
11             {
12               "id": "2AaJeBEq3WLcfFW1y8svDf",
```

```
13              "name": "By Your Side",
14              "album": {
15                "id": "2RrZgDND03MLu6pRJdTkz5",
16                "name": "By Your Side"
17              },
18              "artists": [
19                {
20                  "id": "45TgXXqMDdF8BkjA83OM7z",
21                  "name": "Rod Wave"
22                }
23              ],
24              "features": {
25                "danceability": 0.649,
26                "energy": 0.508,
27                "key": 8,
28                "loudness": -10.232,
29                "mode": 1,
30                "speechiness": 0.0959,
31                "acousticness": 0.0345,
32                "instrumentalness": 3.59e-05,
33                "liveness": 0.0736,
34                "valence": 0.405,
35                "tempo": 157.975,
36                "duration_ms": 194051,
37                "time_signature": 4
38              }
39            },
40            ...
```

This JSON contains all the necessary fields that are needed for continueing with the CRISP DM process. To prepare the dataset for data analysis, the JSON data needs to be transformed into a flat structure to be stored as a CSV file. Another Python function is used to flatten the data. It takes the resulting data from the last step and transforms it into the following format.

```
1  [
2    {
3        "categories.id": "hiphop",
4        "categories.name": "Hip-Hop",
5        "categories.playlists.id": "37i9dQZF1DX0XUsuxWHRQd",
6        "categories.playlists.name": "RapCaviar",
7        "categories.playlists.tracks.id": "2AaJeBEq3WLcfFW1y8svDf",
8        "categories.playlists.tracks.name": "By Your Side",
9        "categories.playlists.tracks.album.id": "2RrZgDND03MLu6pRJdTkz5",
10       "categories.playlists.tracks.album.name": "By Your Side",
11       "categories.playlists.tracks.artists": "Rod Wave",
12       "categories.playlists.tracks.features.danceability": "0.649",
13       "categories.playlists.tracks.features.energy": "0.508",
14       "categories.playlists.tracks.features.key": "8",
15       "categories.playlists.tracks.features.loudness": "-10.232",
16       "categories.playlists.tracks.features.mode": "1",
17       "categories.playlists.tracks.features.speechiness": "0.0959",
18       "categories.playlists.tracks.features.acousticness": "0.0345",
19       "categories.playlists.tracks.features.instrumentalness": "3.59e-05",
```

```
20        "categories.playlists.tracks.features.liveness": "0.0736",
21        "categories.playlists.tracks.features.valence": "0.405",
22        "categories.playlists.tracks.features.tempo": "157.975",
23        "categories.playlists.tracks.features.duration_ms": "194051",
24        "categories.playlists.tracks.features.time_signature": "4"
25    },
26    ...
27 ]
```

Notice, that the nested fields and arrays have been flattened to represent a two dimensional data structure, containing only one array to represent rows which contains json objects with 22 fields. Each field represents a column. In the original JSON data, "artists" is a list containing one or more entries. If the data was flattened in a way that groups the entries by each individual artist, there could be multiple rows for the same track. If for example track1 had two artists artist1 and artist2, there would be two datapoints, both containing the same song name, id and features, but different artists. As the dataset should be grouped by individual tracks, not by artists, which results in multiple entries per track, multiple artists are condensed into a comma-seperated list during flattening. This list is stored in the "categories.playlists.tracks.artists" field. This means, that the individual artist data is lost in favor of grouping the dataset by tracks.

After this process, the resulting two-dimensional structure is converted to a CSV file and saved for further processing.

3.2 Data Understanding

3.3 Data Preparation

In the following section the data collected during data collection is prepared and cleaned up for training the model. At the end of data collection, the result was saved in a CSV file. This file is imported and converted into a Pandas Dataframe using the Pandas Python library and it's "read_csv" method:

```
1 import os
2 import pandas as pd
3
4 # reading data from csv
5 df = pd.read_csv(os.path.join('..', 'data_collection', 'final_result.csv'))
```

Next, duplicates are removed. If there are two or more datapoints, which have the same value in their artist and name column, only the first one is kept and all subsequent entries are removed. This deduplication could also be done by using track ids. The drawback of

this method is, that many artists release their tracks multiple times, e.g. as a single and later in an album. These duplicates would not be caught using the id, as different releases have different id values. As the artist and title doesn't change, almost all duplicates are caught using artist and name.

```
1   df = df.drop_duplicates(subset=['categories.playlists.tracks.artists', 'categories
        .playlists.tracks.name'])
```

Next, all genres that are not to be used for training the model are filtered out.

```
1   genre_filter = ['hiphop', 'jazz', 'rock']
2   df = df[df['categories.id'].isin(genre_filter)]
```

Then columns that are not needed for training are removed, including category name, all playlist information and all track and album information. The category id is kept, as it will serve as the label. The remaining columns are renamed, as the long JSON tree names are no longer needed. The field "category.id" is renamed to "category" and each audio feature is renamed for example the danceability column is now called "feature_danceability".

```
1   columns_to_drop = [
2       "categories.name",
3       "categories.playlists.id",
4       "categories.playlists.name",
5       "categories.playlists.tracks.id",
6       "categories.playlists.tracks.name",
7       "categories.playlists.tracks.album.id",
8       "categories.playlists.tracks.album.name",
9       "categories.playlists.tracks.artists"
10      ]
11  df = df.drop(columns=columns_to_drop)
12
13  df = df.rename(columns={
14      "categories.id": "category",
15      "categories.playlists.tracks.features.danceability": "feature_danceability",
16      "categories.playlists.tracks.features.energy": "feature_energy",
17      "categories.playlists.tracks.features.key": "feature_key",
18      "categories.playlists.tracks.features.loudness": "feature_loudness",
19      "categories.playlists.tracks.features.mode": "feature_mode",
20      "categories.playlists.tracks.features.speechiness": "feature_speechiness",
21      "categories.playlists.tracks.features.acousticness": "feature_acousticness",
22      "categories.playlists.tracks.features.instrumentalness": "
            feature_instrumentalness",
23      "categories.playlists.tracks.features.liveness": "feature_liveness",
24      "categories.playlists.tracks.features.valence": "feature_valence",
25      "categories.playlists.tracks.features.tempo": "feature_tempo",
26      "categories.playlists.tracks.features.duration_ms": "feature_duration_ms",
27      "categories.playlists.tracks.features.time_signature": "feature_time_signature
            "
28  })
29  df
```

With unnecessary columns removed, a check is done to show any remaining null values.

```
1    df.isnull().sum()
```

In this dataset, there are no null values present. As the dataset is now clean, it can now be prepared for input into sklearn's Gradient Boosting Classifier, which is used for training. The classifier only supports integer values as labels, which means that the category data needs to be encoded. This is done by defining a function "encode_target", which takes the dataframe and the name of the column containing the target. It then maps an integer to each unique value in the target column, adds a new "target" column in the dataframe and stores the mapped integer in it.

```python
1    # sklearn takes the features and labels as seperate lists
2    # df needs to be split
3    def encode_target(df, target_column):
4
5        df_mod = df.copy()
6        map_to_int = {name: n for n, name in enumerate(df_mod["category"].unique())}
7        df_mod["target"] = df_mod[target_column].replace(map_to_int)
8
9        return (df_mod)
10
11   df_target = encode_target(df, "category")
```

The head of the resulting dataframe is shown in table 1. The category to target integer mapping is shown in table 2.

Table 1: Dataframe after cleanup

category	feature_danceability	feature_energy	feature_key	feature_loudness	feature_mode
hiphop	0.649	0.508	8.0	-10.232	1.0

Table 2: Category to target integer mapping

target	category
0	hiphop
1	rock
2	jazz

Now that all columns needed for training the model are prepared, the dataset needs to be split in test and training data. This is necessary to be able to calculate an accuracy score for the model after training. In order to calculate this score, data is needed that the model has never seen before, which ensures that the score is not just a result of overfitting, but the accuracy would be the same on real world data. For splitting the data, the function

train_test_split from sklearn is used. First the dataset is shuffled so each set contains a near equal amount of entries for each genre. Then the shuffled dataset is split, with an 80% of samples going into the training set and 20% going into the test set. This results in 10.701 rows of training data and 2.676 rows of test data.

```
from sklearn.model_selection import train_test_split
df_target_shuffled = df_target.sample(frac=1, random_state=45)
train, test = train_test_split(df_target_shuffled, test_size=0.2, random_state=45,
    shuffle=False)
```

Next, features and labels are split into seperate datasets. This is necessary, because the GradientBoostingClassifiers fit method takes features and labels as seperate arguments called X for input and y for label. Datapoints are matched by their row numbers. Labels are stored in a list containing all values from the target column like so.

```
# y contains list of target values
y_train = train["target"]
y_test = test["target"]
y_all = df_target_shuffled["target"]
```

The features are stored in dataframes. The columns are selected using a list of all column names which contain features and then creating a new dataset only containing those columns:

```
# columns 1 to 14 contain the features, column 0 is the category and 15 the target
features = list(train.columns[1:14])

# create datasets only containing feature columns
X_train = train[features]
X_test = test[features]
X_all = df_target_shuffled[features]
```

At this point, all data has been cleaned up, targets have been generated, it has been split into training and test data and features have been seperated from labels. The data is now ready for training the model.

3.4 Modeling

3.4.1 The Classifier

For training the Gradient Boosting Algorithm, the GradientBoostingClassifier class from scikit learn is used. It is part of a group of classes offering all sorts of ensemble methods. As explained in {Grundlagenteil Schwabe} ensemble methods combine the predictions of several weak learners to generate a more robust model and reduce overfitting issues.

sklearn supports averaging methods like Bagging and Random Forests, which take the average of each learners prediction as their output. It also supports Boosting Methods like AdaBoost or Gradient Boosting, which build base estimators sequentially improving the output each time. Sklearn also provides a classifier and regressor model for each method.

The Gradient Boosting Classifier supports binary and multi-class classification and uses 20 hyperparameters to control the size of each regression tree, the number of trees, the learning rate and many more. The most impactful of these parameters are explained here.

- **learning_rate**

  As explained in section **??**, the learning rate is used to control how much each tree contributes to the result, by multiplying it with the output values of the previous tree. The default value here is 0.1

- **n_estimators**

  The number of weak learners to be used while boosting. The default is 100.

- **max_depth**

  The maximum depth of each regression tree. This also impacts its number of nodes. The default value is 3.

- **loss**

  The type of loss function to be used. Possible values are deviance and exponential. Deviance describes how well a logistic regression model, in our case the regression tree, fits the data. A deviance of 0 would be a perfect fit, the higher the number, the worse the fit.

- **criterion**

- **min_samples_split**

- **min_samples_leaf**

- **max_features**

3.4.2 Cross Validation

{explain cross validation here}

3.4.3 Hyperparameter Tuning

Sklearn also provides methods for Hyperparameter tuning. In this project, Exhaustive Grid Search was used to tune hyperparameters. To perform this search, the class Grid-SearchCV can be used. It takes an estimator, in our case the Gradient Boosting Classifier, and a grid of parameters. In this case, the hyperparameters are the parameters of GradientBoostingClassifier as explained above. This codeblock shows a parameter grid, as it is defined in this project.

```
param_grid = {
    "loss": ["deviance", "exponential"],
    "n_estimators": [97, 98, 99, 100, 101, 102, 103],
    "learning_rate": [0.8, 0.9, 0.1, 0.11, 0.12],
    "criterion": ['friedman_mse', 'squared_error'],
    "min_samples_split": [1, 1.5, 2, 2.5, 3],
    "min_samples_leaf": [1],
    "max_depth": [1, 2, 3, 4, 5],
    "random_state": [42],
    "max_features": [None, 0.95, 0.90]
}
```

The parameter grid is given as a Python dictionary containing the parameter name as keys and an array of values for each key. Additionaly a GradientBoostingClassifier object is defined and both are passed into the GridSearchObject.

```
gbc = GradientBoostingClassifier(random_state=45)
search = GridSearchCV(gbc, param_grid)
```

The search object's fit method combines cross-validation, hyperparameter tuning and classifier fitting to find the best possible combination of hyperparameters to the estimator for the given dataset. This is done in the following way:

1. An estimator is created using the first combination of hyperparameters. In the current example, this is equivalent to creating the following estimator.

```
estimator = GradientBosstingClassifier(loss="deviance", n_estimators=97,
        learning_rate=0.8, criterion='friedman_mse', min_samples_split=1,
    min_samples_leaf=1, max_depth=1, random_state=42, max_features=None)
```

2. The newly created classifier is fitted to the training data using this set of parameters using 5-fold cross validation. When this is done, an average of the five scores resulting from cross-validation is built. This average is now used as the score for this specific combination of parameters.

3. The score is saved together with the parameters

4. All of the above steps are repeated with every possible combination of parameters from the parameter grid. This results in a score for every set of parameters.

5. As every model was fitted using the same method, it is clear that the model with the highest score is using the best hyperparameters. A new model is trained without cross-validation so using all data, to create the final model using the best parameters.

The model created by grid search can then be evaluated using test data that was not used in cross validation. Executing this GridSearch in Python is done be calling the fit method of the search object defined in the code above using the train features and labels.

```
1    search.fit(X_train, y_train)
```

After fitting, the search object can be used as if it was a normal estimator built with the optimal hyperparameters. To see, which hyperparameters were optimal, the get_params can be used. It's output looks like this:

```
1    {'cv': 5,
2    'error_score': nan,
3    'estimator__ccp_alpha': 0.0,
4    'estimator__criterion': 'friedman_mse',
5    'estimator__init': None,
6    'estimator__learning_rate': 0.1,
7    'estimator__loss': 'deviance',
8    'estimator__max_depth': 3,
9    'estimator__max_features': None,
10   'estimator__max_leaf_nodes': None,
11   'estimator__min_impurity_decrease': 0.0,
12   'estimator__min_samples_leaf': 1,
13   'estimator__min_samples_split': 2,
14   'estimator__min_weight_fraction_leaf': 0.0,
15   'estimator__n_estimators': 100,
16   'estimator__n_iter_no_change': None,
17   'estimator__random_state': 45,
18   'estimator__subsample': 1.0,
19   'estimator__tol': 0.0001,
20   'estimator__validation_fraction': 0.1,
21   'estimator__verbose': 0,
22   'estimator__warm_start': False,
23   'estimator': GradientBoostingClassifier(random_state=45),
24   'n_jobs': -1,
25   'param_grid': {'loss': ['deviance', 'exponential'],
26       'n_estimators': [97, 98, 99, 100, 101, 102, 103],
27       'learning_rate': [0.8, 0.9, 0.1, 0.11, 0.12],
28       'criterion': ['friedman_mse', 'squared_error'],
29       'min_samples_split': [1, 1.5, 2, 2.5, 3],
30       'min_samples_leaf': [1],
31       'max_depth': [1, 2, 3, 4, 5],
32       'random_state': [42],
33       'max_features': [None, 0.95, 0.9]},
34   'pre_dispatch': '2*n_jobs',
```

```
35    'refit': True,
36    'return_train_score': False,
37    'scoring': None,
38    'verbose': 0}
```

All fields starting with "estimator__" contain the optimal hyperparameters for the data. Even though a broad spectrum of values was given in the search grid, most of the optimal values are equal to the default values given by sklearn. This might suggest, that the developers set the default values using a similar dataset to the one used in this project.

Although many combinations were evaluated, there is still a possibility, that this is a local maximum and a set of hyperparameters that is far different from the ones tested here are actually optimal. Trying out every possible combination of values would require a great amount of computing resources, as the Grid Search shown above already took more than 1.5 hours on a modern 16 core CPU.

Finally the model can be evaluated using the test set defined previously. This set has never been seen by the model even while doing cross validation.

```
1    search.score(X_test, y_test)
```

The estimators score function shows a 86.286% accuracy for the final model using the test set.

3.5 Evaluation

4 Fazit

Appendix

Appendix 1:   Beispielanhang

Dieser Abschnitt dient nur dazu zu demonstrieren, wie ein Anhang aufgebaut seien kann.

Appendix 1.1:   Weitere Gliederungsebene

Auch eine zweite Gliederungsebene ist möglich.

Appendix 2:   Bilder

Auch mit Bildern. Diese tauchen nicht im Abbildungsverzeichnis auf.

Figure 9: Beispielbild

Bibliography

[1]   A. Meier, 'Rundgang big data analytics – hard & soft data mining,' in *Big Data Analytics*, Springer Fachmedien Wiesbaden, 2021, pp. 3–23. DOI: 10.1007/978-3-658-32236-6_1.

[2]   Chen, Chiang, and Storey, 'Business intelligence and analytics: From big data to big impact,' *MIS Quarterly*, vol. 36, no. 4, p. 1165, 2012. DOI: 10.2307/41703503.

[3]   M. Tanwar, R. Duggal, and S. K. Khatri, 'Unravelling unstructured data: A wealth of information in big data,' in *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, IEEE, 2015-09. DOI: 10.1109/icrito.2015.7359270.

[4]   D. Fasel and A. Meier, 'Was versteht man unter big data und NoSQL?' In *Big Data*, Springer Fachmedien Wiesbaden, 2016, pp. 3–16. DOI: 10.1007/978-3-658-11589-0_1.

[5]   'Quick guide: Machine learning im maschinen und anlagenbau,' VDMA Software und Digitalisierung, 2018.

[6]   S. Schacht and C. Lanquillon, *Blockchain und maschinelles Lernen: Wie das maschinelle Lernen und die Distributed-Ledger-Technologie voneinander profitieren*. Springer Berlin Heidelberg, 2019, ISBN: 9783662604083. [Online]. Available: https://books.google.de/books?id=nL%5C_ADwAAQBAJ.

[7]   G. Paaß and D. Hecker, *Künstliche Intelligenz*. Springer Fachmedien Wiesbaden, 2020. DOI: 10.1007/978-3-658-30211-5.

[8]   P. Buxmann and H. Schmidt, *Künstliche Intelligenz: Mit Algorithmen zum wirtschaftlichen Erfolg*. Springer Berlin Heidelberg, 2018, ISBN: 9783662575680. [Online]. Available: https://books.google.de/books?id=2f1sDwAAQBAJ.

[9]   H. Ernst, J. Schmidt, and G. Beneken, *Grundkurs Informatik: Grundlagen und Konzepte für die erfolgreiche IT-Praxis - Eine umfassende, praxisorientierte Einführung*. Springer Fachmedien Wiesbaden, 2016, ISBN: 9783658146344. [Online]. Available: https://books.google.de/books?id=hiLFDAAAQBAJ.

[10]   W. Ertel, *Grundkurs Künstliche Intelligenz*. Springer Fachmedien Wiesbaden, 2021. DOI: 10.1007/978-3-658-32075-1.

[11]   'Maschinelles lernen: Eine analyse zu kompetenzen, forschung und anwendung,' Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V, Tech. Rep., 2018.

[13]   G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, 2nd ed. Springer US, 2021. DOI: 10.1007/978-1-0716-1418-1.

[14]   R. J. Lewis, 'An introduction to classification and regression tree (cart) analysis,' in *Annual meeting of the society for academic emergency medicine in San Francisco, California*, Citeseer, vol. 14, 2000.

[15]   J. H. Friedman, 'Greedy function approximation: A gradient boosting machine.,' *The Annals of Statistics*, vol. 29, no. 5, 2001-10. DOI: 10.1214/aos/1013203451.

[16]   A. Natekin and A. Knoll, 'Gradient boosting machines, a tutorial,' *Frontiers in Neuro-robotics*, vol. 7, 2013. DOI: 10.3389/fnbot.2013.00021.

[17]   M. Reddy, *API Design for C++*. Elsevier Science, 2011, ISBN: 9780123850041. [Online]. Available: https://books.google.de/books?id=IY29LylT85wC.

[20]   J. Hemming, *Methoden der Erforschung populärer Musik*. Gabler, Betriebswirt.-Vlg, 2015-10-28, 514 pp., ISBN: 3658114967. [Online]. Available: https://www.ebook.de/de/product/25204785/jan_hemming_methoden_der_erforschung_populaerer_musik.html.

Internet sources

[12] L. Wuttke. 'Machine learning: Definition, algorithmen, methoden und beispiele,' data-solut. (), [Online]. Available: https://datasolut.com/was-ist-machine-learning/#teilueberwachtes-lernen (visited on 2022-01-23).

[18] R. Becker. 'What is music?' Study.com. (2021-10-29), [Online]. Available: https://study.com/learn/lesson/what-is-music-characteristics-of-music-how-music-is-made.html.

[19] R. Havers. 'John coltrane quotes: The iconic saxophonist in his own words,' UDiscoverMusic. (2021-09-23), [Online]. Available: https://www.udiscovermusic.com/stories/john-coltrane-in-20-quotes/.

[21] 'Chapter 2: Music: Fundamentals and educational roots in the u.s.' (), [Online]. Available: https://milnepublishing.geneseo.edu/music-and-the-child/chapter/chapter-2/.

[22] 'Music theory: 5 fundamentals that you should know,' Shaw Academy. (2019-04), [Online]. Available: https://www.shawacademy.com/blog/music-theory-5-fundamentals-that-you-should-know/.

[23] M. Shipman. 'Analysis of 50 years of hit songs yields tips for advertisers,' NC State University. (2014-03-18), [Online]. Available: https://news.ncsu.edu/2014/03/wms-henard-hits2014/.

[24] 'What are we listening to? | the most popular themes in songs,' Atlanta Institute of Music and Media. (2020-02-14), [Online]. Available: https://www.aimm.edu/blog/most-popular-themes-in-songs.

[25] 'Instrument classification,' Goshen College. (), [Online]. Available: https://www.goshen.edu/academics/music/mary-k-oyer-african-music-archive/instrument-classification/.

[26] 'Musikepochen,' musiklexikon.info. (), [Online]. Available: http://www.musiklexikon.info/musiklexikon/musikepochen.

[27] 'Musikrichtungen,' MUSICFLX. (), [Online]. Available: https://www.musicflx.de/musikrichtungen/.

[28] 'What is hip-hop?' Musical Dictionary. (), [Online]. Available: https://musicaldictionary.com/what-is-hip-hop/.

[29] G. Tate. 'Hip-hop music and cultural movement,' Encyclopædia Britannica, Inc. (), [Online]. Available: https://www.britannica.com/art/hip-hop.

[30] R. PQ. 'Hip hop history: From the streets to the mainstream,' Icon Collective. (2019-11-13), [Online]. Available: https://iconcollective.edu/hip-hop-history/.

[31] K. Goodrich. 'What is hip-hop?' (2017-04-25), [Online]. Available: https://medium.com/@katiegoodrich/what-is-hip-hop-9f9abfffb25e.

[32] M. Beek. 'What is jazz music?' BBC Music Magazine. (2021-07-15), [Online]. Available: https://www.classical-music.com/features/articles/jazz-music-what-it-is-and-how-it-evolved/.

[33] D. J. Wildridge. 'Characteristics of jazz: An introduction,' CMUSE. (2020-01-15), [Online]. Available: BBC%20Music%20Magazine.

[34] 'What is jazz?' National Museum of American History. (), [Online]. Available: https: //americanhistory.si.edu/smithsonian-jazz/education/what-jazz.

[35] 'What is jazz? a guide to the history and sound of jazz,' MasterClass. (2020-11-08), [Online]. Available: https://www.masterclass.com/articles/what-is-jazz#what-is-jazz-music.

[36] 'What is jazz?' Musical Dictionary. (), [Online]. Available: https://musicaldictionary. com/what-is-jazz/.

[37] 'Rock 'n' roll music guide: 4 characteristics of rock 'n' roll,' MasterClass. (2021-10-25), [Online]. Available: https://www.masterclass.com/articles/rock-n-roll-music-guide.

[38] 'What is rock music?' Musical Dictionary. (), [Online]. Available: https : / / musicaldictionary.com/what-is-rock-music/.

[39] B. Clark. '30 different types of rock music (rock subgenres),' MusicianWave.com. (2021-08-19), [Online]. Available: https://www.musicianwave.com/types-of-rock-music-subgenres/.

[40] 'Spotify web api documentation,' Spotify AB. (), [Online]. Available: https://developer. spotify.com/documentation/web-api/.

[41] 'What is postman?' Postman, Inc. (), [Online]. Available: https://www.postman.com/ product/what-is-postman/ (visited on 2022-01-16).

[42] 'Spotify authorization documentation,' Spotify AB. (), [Online]. Available: https :// developer.spotify.com/documentation/general/guides/authorization/.

Declaration in lieu of oath

I hereby declare that I produced the submitted paper with no assistance from any other party and without the use of any unauthorized aids and, in particular, that I have marked as quotations all passages which are reproduced verbatim or near-verbatim from publications. Also, I declare that the submitted print version of this thesis is identical with its digital version. Further, I declare that this thesis has never been submitted before to any examination board in either its present form or in any other similar version. I herewith agree/disagree that this thesis may be published. I herewith consent that this thesis may be uploaded to the server of external contractors for the purpose of submitting it to the contractors' plagiarism detection systems. Uploading this thesis for the purpose of submitting it to plagiarism detection systems is not a form of publication.

Düsseldorf, 23.1.2022

(Location, Date)   (handwritten signature)