



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	IV
List of Tables	V
List of Abbreviations	VI
List of Symbols	VII
1 Einleitung	1
1.1 Problem Definition	1
1.2 Goal	1
1.3 Structure	1
2 Fundamentals	2
2.1 Basic Concepts of Big Data	2
2.1.1 Relevance of Data	2
2.1.2 The 5V Matrix for Big Data	3
2.2 Differencation between Big Data and Big Data Analytics	3
2.3 Data Formats	4
2.3.1 Reinforcement Learning	5
2.3.2 Machine Learning Algorithms	5
2.4 Decision Trees	5
2.4.1 Decision Tree Algorithm	6
2.4.2 Evaluation of Decision Trees	9
2.5 Gradient Boosting	9
2.5.1 Gradient Boosting Algorithm	10
2.5.2 Evaluation of Gradient Boosting	13
2.6 Cross Industry Standard Process for Data Mining	14
2.7 Application Programming Interfaces	14
2.7.1 Purpose and Usage	14
2.8 Basic Concepts of Music Theory	14
3 Implementation	15
3.1 Data Collection	15
3.1.1 Requirements for the dataset	15
3.1.2 Existing Datasets	16
3.1.3 Ressources and Approach	16

3.1.4	Authorization	17
3.1.5	Getting Features	19
3.1.6	Getting Track IDs and Labels	21
3.2	Data Understanding	24
3.3	Data Preparation	24
3.4	Modeling	24
3.5	Evaluation	24
4	Fazit	24
	Appendix	25
	Bibliography	26

List of Figures

Figure 1: Spotify Authorization Flow	18
Figure 3: Audio Feature Request	20
Figure 4: Artist Request	22
Figure 5: Categories and Playlists in Spotify App	23

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 its 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 Machine Learning Algorithms

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 [5, 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 [6, 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 1. 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 splitting 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**.

$$\theta^* = \operatorname{argmin}_{\theta} G(Q_m, \theta)$$

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 [5, 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 .

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

(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 [6, 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 [6, 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 [5, 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 N_m 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 universally 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 [5, 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 memorization but a comparatively weak generalization ability [5, 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 again be tested against the subset that is reserved for validation with the best model chosen [6, 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 [5, 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 Trees. 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 dependence 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) [7, p.1189] [8, 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 [8, 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 Learners to choose from, again the choice is mostly depending on the prediction task and available data [8, 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 a 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 x_1 -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 $\log(\text{odds})$ -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”.

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

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 $\log(\text{odds})$ -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 out 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 [7, p.1195].

FIGURE REGRESSION 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 (13). 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 $\log(\text{odds})$ -prediction is based on the last $\log(\text{odds})$ -prediction plus the Learning Rate v multiplied by the Output Value(s) for the sample of the last Regression Tree (14) [7, p.1203]. Normally, there is only one Output Value for a sample which makes

the summation sign obsolete. The Learning Rate ν is a hyperparameter for Gradient Boosting. For this example, a high ν is used to better visualize the changes. In practice a ν in the order of 0.1 is common as decreasing the Learning Rate tends to give better results [7, p.1206].

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

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 predictions must again be transformed to probabilities like for the calculation of the PR. For the final probabilities thresholds are used to compute the category to which the samples belong [7, 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 capture 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 flexibility 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 evaluation 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 prediction is better, it is much more complex to evaluate the model and explain the results [8, 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[9, 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.[9, 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

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 include 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>

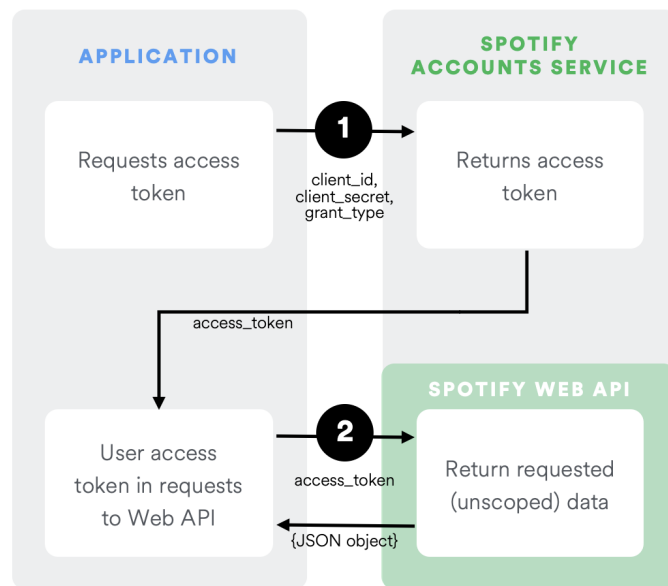
⁶ <https://developer.spotify.com/>

The API is based on the Representational State Transfer (REST) architecture. The different endpoints return JSON metadata directly from the Spotify Data Catalogue [10]. 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. [10] 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. [11] 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.

3.1.4 Authorization

Using the Spotify documentation for authorization workflows [12], 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 from the API. 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: [12]

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 ?? shows the full request and response to acquire the token.

Figure 2: Access Token Request

POST	https://accounts.spotify.com/api/token <i>request access token</i>
Body	application/x-www-form-urlencoded
	<pre> 1 { 2 "grant_type": "client_credentials", 3 "client_id": client id from application dashboard, 4 "client_secret": client secret from dashboard 5 }</pre>

Response	application/json
200 ok	<pre> 1 { 2 "access_token": "BQDgQCSx-tIMDo9LfVeZxm6Ym12p_WbEU3Q 9ENsVl7e--6d_vockTsfzMVUhPWihSSnFUuHvm_9POA1kYEw" , 3 "token_type": "Bearer", 4 "expires_in": 3600 5 }</pre>

In the Python implementation, the requests library is used to execute the request in figure ?? and store the access token in a variable. The dotenv library is used to read the client id and secret from a separate .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.

```

1  #Get environment variables from ".env" file and read credentials
2  load_dotenv('.env')
3  client_id = os.environ.get('CLIENT_ID')
4  client_secret = os.environ.get('CLIENT_SECRET')
5
6  # Authenticate and get an API Token from Spotify using a Client ID and secret
7  def getAuthTokenFromCredentials(id, secret):
8
9      url = "https://accounts.spotify.com/api/token"
10
11      payload = f'grant_type=client_credentials&client_id={id}&client_secret={secret}'
12
13      headers = {
14          'Content-Type': 'application/x-www-form-urlencoded',
15      }
16
17      response = requests.request("POST", url, headers=headers, data=payload)
18
19      return response.json()["access_token"]
20
21  auth_token = getAuthTokenFromCredentials(client_id, client_secret)
```

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 50 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-request endpoint of a single track is shown in figure 3.

Figure 3: Audio Feature Request

GET	https://api.spotify.com/v1/audio-features/{id} <i>request audio features for id</i>
Parameter	
id	id of the song
Response	application/json
200	ok
<pre> 1 { 2 "danceability": 0.677, 3 "energy": 0.638, 4 "key": 8, 5 "loudness": -8.631, 6 "mode": 1, 7 "speechiness": 0.333, 8 "acousticness": 0.589, 9 "instrumentalness": 0, 10 "liveness": 0.193, 11 "valence": 0.435, 12 "tempo": 82.810, 13 "type": "audio_features", 14 "id": "2e3Ea0o24lReQFR4FA7yXH", 15 "uri": "spotify:track:2e3Ea0o24lReQFR4FA7yXH", 16 "track_href": "https://api.spotify.com/v1/tracks/2e3Ea0o24lReQFR4FA7yXH", 17 "analysis_url": "https://api.spotify.com/v1/audio-analysis/2e3Ea0o24lReQFR4FA7yXH", 18 "duration_ms": 211497, 19 "time_signature": 4 20 }</pre>	

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.

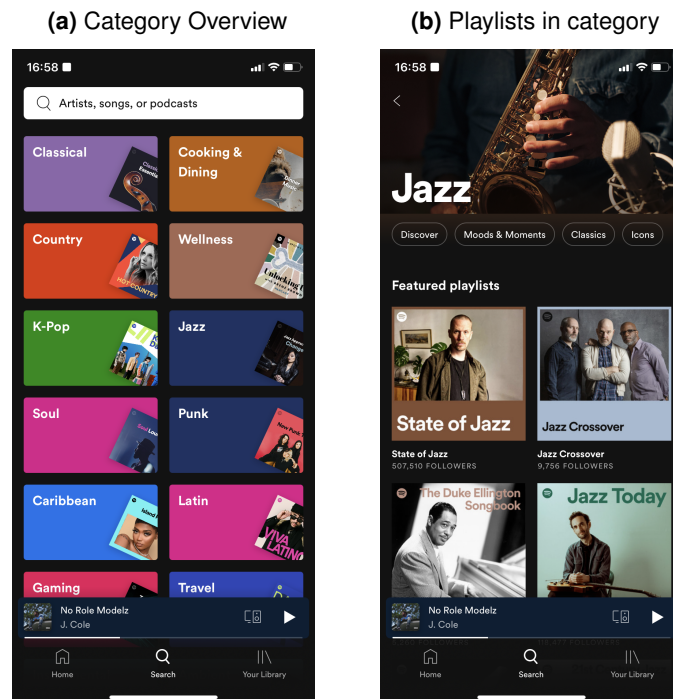
Figure 4: Artist Request

GET	https://api.spotify.com/v1/artists/{id} <i>request information about an artist by their id</i>
Parameter	
id	id of the artist
Response	
application/json	
200	ok
<pre> 1 { 2 "external_urls": { 3 "spotify": "https://open.spotify.com/artist/6l3HvQ5sa6mXTsMTB19rO5" 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/6l3HvQ5sa6mXTsMTB19rO5", 16 "id": "6l3HvQ5sa6mXTsMTB19rO5", 17 "images": [18 ... 19], 20 "name": "J. Cole", 21 "popularity": 89, 22 "type": "artist", 23 "uri": "spotify:artist:6l3HvQ5sa6mXTsMTB19rO5" 24 }</pre>	

This exemplary 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 Spotify's "categories" feature. The app's 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. The API mirrors the app's 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.

Figure 5: Categories and Playlists in Spotify App



3.2 Data Understanding

3.3 Data Preparation

3.4 Modeling

3.5 Evaluation

4 Fazit

Appendix

Appendix 1: Beispielanhang

Dieser Abschnitt dient nur dazu zu demonstrieren, wie ein Anhang aufgebaut sein 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 6: Beispielbild

Name	Änderungsdatum	Typ	Größe
 abbildungen	29.08.2013 01:25	Dateiordner	
 kapitel	29.08.2013 00:55	Dateiordner	
 literatur	31.08.2013 18:17	Dateiordner	
 skripte	01.09.2013 00:10	Dateiordner	
 compile.bat	31.08.2013 20:11	Windows-Batchda...	1 KB
 thesis_main.tex	01.09.2013 00:25	LaTeX Document	5 KB

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, Info-com 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] 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.
- [6] 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.
- [7] 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.
- [8] A. Natekin and A. Knoll, 'Gradient boosting machines, a tutorial,' *Frontiers in Neuro-robotics*, vol. 7, 2013. DOI: 10.3389/fnbot.2013.00021.
- [9] M. Reddy, *API Design for C++*. Elsevier Science, 2011, ISBN: 9780123850041. [Online]. Available: <https://books.google.de/books?id=IY29LyIT85wC>.

Internet sources

- [10] 'Spotify web api documentation,' Spotify AB. (), [Online]. Available: <https://developer.spotify.com/documentation/web-api/>.
- [11] 'What is postman?' Postman, Inc. (), [Online]. Available: <https://www.postman.com/product/what-is-postman/> (visited on 2022-01-16).
- [12] '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)

A handwritten signature in black ink, consisting of a large, stylized 'H' followed by a series of loops and a final flourish.

(handwritten signature)