UPPSALA
UNIVERSITET

UPTEC STS 19007
Examensarbete 30 hp
Mars 2019

Language Classification of Music Using Metadata

Linus Roxbergh

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Language Classification of Music Using Metadata

Linus Roxbergh

The purpose of this study was to investigate how metadata from Spotify could be used to identify the language of songs in a dataset containing nine languages. Features based on song name, album name, genre, regional popularity and vectors describing songs, playlists and users were analysed individually and in combination with each other in different classifiers. In addition to this, this report explored how different levels of prediction confidence affects performance and how it compared to a classifier based on audio input.

A random forest classifier proved to have the best performance with an accuracy of 95.4% for the whole data set. Performance was also investigated when the confidence of the model was taken into account, and when only keeping more confident predictions from the model, accuracy was higher. When keeping the 70% most confident predictions an accuracy of 99.4% was achieved. The model also proved to be robust to input of other languages than it was trained on, and managed to filter out unwanted records not matching the languages of the model. A comparison was made to a classifier based on audio input, where the model using metadata performed better on the training and test set used. Finally, a number of possible improvements and future work were suggested.

Handledare: Jan Stypka (Spotify)
Ämnesgranskare: Niklas Wahlström
Examinator: Elísabet Andrésdóttir
ISSN: 1650-8319, UPTEC STS 19007

Populärvetenskaplig sammanfattning

I takt med utvecklingen av datorer och internet så har sättet som människor konsumerar musik ändrats drastiskt. Streamingtjänster som Spotify, Apple Music och Pandora ger användare tillgång till miljoner låtar en knapptryckning bort, till skillnad från tidigare då musik behövde köpas genom fysiska kopior över disk. Till följd av denna utveckling så har ett ökat fokus på att personalisera musikupplevelsen till varje användare blivit vanligare och vanligare. Som en del i att personalisera upplevelsen så har språket i låtar blivit en viktig faktor. Ett exempel som illustrerar hur språket i musik kan vara avgörande var då artisten AR Rahman spelade på Wembley i London 2017. Rahman sjöng en majoritet av låtarna på Tamil, vilket resulterade i en stor del av den Hindi-talande delen av publiken att lämna konserten och kräva pengarna tillbaka. För en stor streaming-tjänst som Spotify så har vikten av att ta hänsyn till språk blivit allt större med en växande katalog av musik för en allt mer mångfaldig användarbas från hela världen.

Syften med den här rapporten var att undersöka hur metadata från Spotify kan användas för att identifiera språk hos låtar i ett dataset innehållande nio språk. Olika typer av data innehållande namn på låtar och albums, genre, popularitet i olika regioner samt information om användare, låtar och spellistor analyserades individuellt och i kombination med varandra i olika klassificerare. Flera av de undersökta datakällorna visade sig innehålla relevant information som kunde användas i klassificerarna, men avsaknad på metadata hos många av låtarna var problematiskt. När alla låtar i det valda datasettet klassificerades uppnåddes en korrekthet på 95.4% och då de 70% mest konfidenta klassificeringarna behölls uppnåddes en korrekthet på 99.4%. En jämförelse med en ljud-baserad klassificerare genomfördes också där modellen baserad på metadata presterade bättre. Avslutningsvis presenteras flera förbättringsområden och möjligheter för framtida forskningsområden.

Acknowledgements

This study is a result of a Master's Thesis research project at Uppsala university, performed in the Growth Opportunities tribe at Spotify. I would like to thank my supervisor at Spotify Jan Stypka for valuable support and help, and Simon Durand and Martin Ståhlgren at Spotify for their input along the way. Finally, I want to thank my university subject reader Niklas Wahlström for valuable insights and pointers.

Contents

1. Introduction

1.1. Background

Since the introduction of computers and the internet, the music industry has been evolving to change how many people access and discover new music [1]. Streaming services such as Spotify, Apple Music and Pandora gives users instant access to millions of tracks, compared to previous times where people bough physical copies of music in form of LPs, cassettes and CDs. With the increasing amount of easily accessible tracks, recommendation systems to help users discover music they like have become more popular in recent research [2, 3, 4]. An example of an implemented recommendation system is Spotify's playlist "Discover Weekly" which applies machine learning to listening habits and playlists to give each user 30 personalized tracks every week.

As a part of the emerging focus of adapting the listening experience for the individual user, the language of tracks has become more important. Language consistency can in many cases be useful, where tracks in different languages are appropriate for different users. For example, a person in Argentina might enjoy tracks in Spanish more than those in Danish or Afrikaans. An example illustrating the importance of language in music was during the Oscar-award winning musician AR Rahman's concert in Wembley in 2017. Rahman had sung most of the songs in Tamil, which resulted in a large portion of the Hindi-speaking crowd to leave and demand a refund as they expected him to sing the Hindi version of the songs [5].

Spotify is a music streaming platform launched in 2008 with currently around 40 million unique tracks and 200 million users [6]. For a large streaming platform like Spotify to be able to adapt to languages in certain use cases, the presence of language information for tracks is essential. With a large catalogue that is constantly growing in a rapid pace, manual language tagging of these tracks is unfeasible. Audio classifiers have proved useful in identifying the language of

audio sources [7], but largely depends on the type of music and how distinctive the vocal parts are [8, 9].

Metadata can be defined as data that describes other data, and with Spotify's large collection of metadata, many of these could potentially be a good indicator of what language a track is in. Metadata about where in the world tracks are played the most, in what genres they are in, what types of users that listens to them are examples of available data that could be used. By using these features in machine learning based classification models a scalable solution to label songs with language could potentially be achieved.

1.2. Problem definition

The aim of this report is to investigate the possibility of using metadata at Spotify to predict the language of songs. The goal is to evaluate the predictive power of different features and compare different machine learning models. In addition to this, the report aims to explore how different levels of prediction confidence affects performance and how it compares to a classifier based on audio input.

1.3. Thesis disposition

This report consists of 9 sections. After this paragraph follows section 2 that introduces various concepts, models and methods used in the report, and a section where related work is presented. Section 3 goes through the methodology used and limitations to the report. In section 4 the data sources are presented and the process of selecting, extracting and processing the different features. Section 5 displays the results of the completed experiments followed by section 6 where the results are discussed. In section 7 possible improvements are presented along with potential future work. Finally, section 8 contains a conclusion and summary before the appendix.

2. Theory

This section provides a theoretical background and serves as a basis for the concepts that are utilized throughout the report. Beside the theoretical background, previous related work to language classification and use of metadata is presented.

2.1. Machine learning and classification

Machine learning can be described as the scientific study of algorithms and statistical models used by computers to perform specific tasks without being explicitly programmed, but instead relying on models and inference. The machine learning algorithms build mathematical models of training data to make predictions or decisions for a task without being explicitly programmed for it. Tasks where the training data consists of input vectors with corresponding labels and where the aim is to assign these labels to new data points based on their input vectors are called classification problems [10]. Each element of the input vector are sometimes referred to as *features*.

Multi-class prediction is where the input is to be classified into one of non-overlapping classes [11]. Similar to regression models, classification models can produce a continuous valued prediction, usually in a form of probability ranging from 0 to 1. In addition to the continuous value, it generates the predicted class as a discrete non-overlapping category [12]. Depending on the use case, the information that comes with the probability scores for each record can be used, or directly use the predicted discrete classes. For a classifier used in filtering spam e-mails, a probability of spam of 0.51 and 0.99 for an e-mail would be treated the same if only the discrete categories were used without regard to probabilities. Due to the consequences of classifying non-spam as spam, adjustments for the probability score could be used to only pick predictions where the model is more confident [12].

2.2. Classification models

2.2.1. K-nearest neighbors

The k-nearest neighbors (k-NN) is a model that is widely used in both in unsupervised clustering and supervised classification tasks. It is a non-linear and non-parametric model where the distances from a new data point's input vector to the previous data points input vectors are measured. The new data point is classified with a label in accordance with a majority voting among the k closest neighbours labels [13]. An example with $k = 1$ and $k = 3$ with two classes is shown in figure 1. With $k = 1$ the new entity is classified as a *Class 1* as it is the closest item and with $k = 3$ a majority vote results in it being classified as *Class 2*.

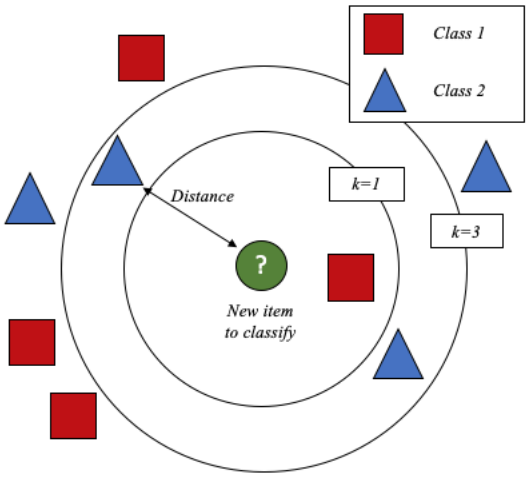

Figure 1: A k-nearest neighbors with $k = 1$ and $k = 3$ is illustrated where the squares and triangles represents two different classes. With $k = 1$ the new item will be classified as *Class 1* and with $k = 3$ as *Class 2*.

There are different ways to measure the distance between points, where the Euclidean distance is the most common measurement. In larger data sets, the search for k-nearest neighbors can be very time consuming [14].

2.2.2. Logistic regression

Logistic regression is a frequently used model in data analysis with a benefit that it is relatively easy to study individual variables effect on the result in the model. The posterior probability P of the classes are being modelled as linear functions of the variables to output a binary class [13]. This can be seen in equation 1 where X_i represents the values of features with corresponding β_i resulting in the posterior probability P for the binary class.

$$P = \frac{1}{1 + \exp[-(1 + \beta_0 + \beta_1 X_1 + ... + \beta_p X_p)]} \tag{1}$$

The parameters of the logistic regression models are usually fit using maximum likelihood. When adapting logistic regression to classification tasks with more than two classes, a common approach is to use a one-vs-rest methodology where a binary problem is fitted for each class [15].

2.2.3. Decision trees

Tree-based models partitions the feature space into different regions, where each region represents a class, and then fit a simple model for each one [13]. Figure 2 illustrates an example with five regions (C_1, C_2, C_3, C_4 and C_5) and two features (X_1 and X_2). The feature space is illustrated on the left-hand side by the rectangles where the thresholds b_1, b_2, b_3, and b_4 separates the different classes. On the right-hand side the tree is visualised with the same thresholds for the two features. This example contains two features but the same concept is applied when using more features.

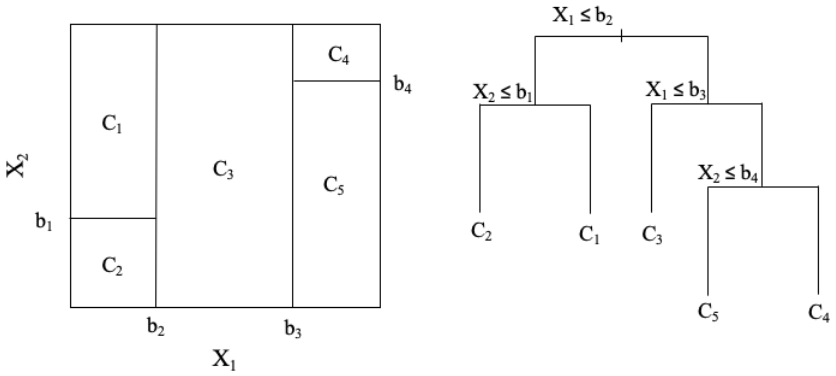

Figure 2: A decision tree with five classes (C_1, C_2, C_3, C_4, C_5) two features (X_1, X_2) and four thresholds (b_1, b_2, b_3, b_4). The left side represents the division of the features space for the five classes and the right side the created decision tree.

The final nodes, also called the leaves of the tree, that end up in the bottom of the decision tree correspond to the different regions or classes C_1 through C_5. In an ideal classification scenario, each node only contain items from the same class. The algorithms for constructing the trees usually use a top-down approach where a variable is chosen at each step to perform the best split of the dataset [16]. There are different algorithms to measure what the best split is, where two popular ones are Gini impurity measure and entropy. To compute Gini impurity and entropy for a set with K classes where $i \in \{1, 2, ..., K\}$ where p_i is the fraction of items labeled with class i in the set, equation 2 is used for Gini and equation 3 for entropy. If all items end the nodes belong to the same class i both Gini impurity and Entropy will be zero.

$$\textit{Gini impurity} = \sum_{i=1}^{K} p_i(1 - p_i) \tag{2}$$

$$\textit{Entropy} = -\sum_{i=1}^{K} p_i \log_2 p_i \tag{3}$$

The complexity of the model depends on the size of the tree, and the optimal size depends on the data used. A large tree could lead to overfitting while a

smaller tree risk missing important structure [13].

2.2.4. Random forest

Bagging is the concept of averaging many noisy but approximately unbiased models to reduce the variance. Decision trees are good candidates for bagging as they can capture complex structures in data and if sufficiently deep, have relatively low bias. Random forest is a modification of bagging that builds a large collection of de-correlated trees to then be averaged for the classification [13]. Several parameters can be adjusted in the random forest classifier, for example the criterion method for splits, the size of the trees and the number of trees. The predicted class for input data is a vote by all the ensembled trees, weighted by their probability estimates. The process can be described as the following:

1. Choose B trees.
2. Use a random sample data from the training set for each tree.
3. Build B random-forest trees by selecting a number of random features and split based on selected criteria until minimum node size is reached.
4. Output the ensemble of trees.
5. For each tree, make a class prediction.
6. Take a majority vote from all B trees predictions to make a final classification [13].

2.2.5. XGBoost

XGboost is short for *extreme gradient boosting* and have some similarities to random forest. XGboost also uses an ensemble of trees, but in addition to this boosting is applied. Boosting is the concept of an iterative method where a new function is added at each step that is based on the error from the previous estimation. In normal gradient boosting the gradient decent is used to minimize the objective function. In XGBoost, both the first and second order gradients

are used on the loss function. XGBoost have become very popular since its initial release in 2014 and have been winning many competition based on structured data problems [17].

2.3. Word2vec

Word2vec is a word embedding method that uses unsupervised training on corpus of text to produce vectors as output. The model represents words found in a vocabulary by a vector of a fixed dimension. With these vectors, the aim is for words that are similar to be close to each other in the vector space, and to enable computations to represent relationships between words [18]. Word vectors which share similar contexts across many documents should end up being neighbors in the vector space. For example, vec("Stockholm") - vec("Sweden") + vec("Norway") would be close to vec("Oslo").

The technique of word2vec is similar to an auto-encoder, but instead of training against input words through reconstruction, it trains words against neighbouring words in the input corpus. It is essentially using a neural network with one input layer, one hidden layer, and one output layer. Word2vec can be implemented in two ways: With Continuous bag of words (CBOW) or Skip-gram. CBOW is using context to predict a target word, where Skip-gram is using a word to predict a target context [19].

Beside the original usage of word2vec to output vectors from corpus, other areas of applications for vector space representation have been used elsewhere. To name a few, it has been used to find item similarities for recommendations by leveraging metadata [20] and in unsupervised feature learning in social networks [21]. At Spotify it is being used to create vectors for tracks, users and playlist to find similarities and build recommendation systems, among other things.

2.4. Missing data

In classification tasks it is not unusual to have observations where values are missing for one or more features [13]. In many cases it can be useful to know

why values are missing. If the pattern of missing data is related to the outcome, that pattern itself can be instructional on its own, which is called *informative missingness* [12]. To handle missing data, there are a few possible approaches:

1. Discard records with any missing values.
2. Use a learning algorithm that deals with missing values in its training phase.
3. Impute all missing values before training.

Discarding records with missing values can be a valid approach in a data set with few missing values, but should otherwise be avoided. While some methods can adapt to missing values, for example XGBoost and CART, a majority would either need to discard records or fill-in missing values. The usual approach is to impute the missing values in some way. A basic tactic is to impute values for the missing features, either based on mean or median values for that feature. An even more simple approach is to impute static values based on assumption from the modeller [13]. Another alternative is to use use a predictive model to estimate the values of the missing values in a more sophisticated way than mean or median [12].

2.5. Evaluation metrics

Evaluation is an important part when building good model for classification. There are different metrics and methods to assess the performance of a model. The importance of these metrics varies from application to application. Except for performance, computational issues and the ease in which the model and assessment can be understood can also be important factors in many applications [22].

2.5.1. Confusion matrix

Confusion matrices are often used to evaluate a classifier. It summarizes the classification performance with respect to selected test data. It is a two-dimensional matrix displaying actual classes on one axis and predictions from the classifier

on the other axis [22]. In a binary case it will be a 2×2 matrix as displayed in table 1 [11].

- True positive *(TP)* - True positives are items of class A that are correctly predicted as items of class A.
- False positive *(FP)* - False positives are items of class B that are incorrectly predicted as items of class A.
- True negative *(TN)* are items of class B that are correctly predicted as items of class B.
- False negative *(FN)* - False negatives are items of class A that are incorrectly predicted as items of class B.

Data class	Classified as positive	Classified as negative
Positive	True positive *(TP)*	False negative *(FN)*
Negative	False positive *(FP)*	True negative *(TN)*

Table 1: A binary confusion matrix displaying the relationship between predicted and actual classes.

For multiple classes the dimension will be k×k for k classes. In a perfect case, all values will be zero except on the diagonal, which indicates correctly classified entities. In table 2 the first row indicates that 8 objects belong to Class A and that 4 are classed correctly as Class A, 1 classified as Class B and 3 as Class C.

Data class	Pred Class A	Pred Class B	Pred Class C
Actual Class A	4	1	3
Actual Class B	2	6	1
Actual Class C	3	2	5

Table 2: A confusion matrix with three classes showing the relationship between prediction and actual classes.

2.5.2. Metrics

Accuracy:

One of the simplest and most straightforward ways of evaluating a classification task is the accuracy rate. It shows the relationship between observed and predicted classes and is fairly intuitive. The number of correct classifications, true positive and true negative, are divided with the total number of predictions. However, accuracy does not make any distinction about the type of errors the classifier makes which is a drawback in some cases [12].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Precision:

Precision can be defined as the number of positive observations that have been correctly classified divided by the total amount of positive predictions. High precision follows a low fraction of false positive classifications.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Recall:

Recall can be described as the correct positive predictions divided by the total number of positive entities. Recall therefore describes the fraction of positive entities that were correctly classifies [12].

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

F-score:

F-score is a evaluation metric commonly used in classification tasks and is a weighted average of precision and recall described previously [23]. This metric combines the precision of the model while also taking into account the fraction of entities classified. The f-score ranges between 0 and 1, where a higher value is better.

$$F_1 = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{7}$$

Macro and micro scores:

In a multi-class setting, it is possible to calculate these scores by using a macro or micro average. A macro mean is based on the performance in each class without regard to the sizes of the classes. For micro averages, the total sum of *TP*, *FP*, *TN* and *FN* are calculated. If the classes are unbalanced a macro score can be very misleading, while a micro score will take the sizes of the classes into account as the total textitTP, *FP*, *TN* and *FN* scores are used [23]. Macro and micro scores for precision and recall are displayed in equations 8 through 11. To calculate a micro f-score, these recall and precision scores are used in equation 7.

$$\text{Macro precision} = \frac{1}{k}\sum_{i=1}^{k}\frac{TP_i}{TP_i + FN_i} \tag{8}$$

$$\text{Micro precision} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} TP_i \sum_{i=1}^{k} FN_i} \tag{9}$$

$$\text{Macro recall} = \frac{1}{k}\sum_{i=1}^{k}\frac{TP_i}{TP_i + FP_i} \tag{10}$$

$$\text{Micro recall} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} TP_i \sum_{i=1}^{k} FP_i} \tag{11}$$

2.5.3. Performance for varied thresholds

In a classifier that outputs a confidence rather than just a classification it is possible to set a threshold to decide what classifications to keep. When using a threshold, it is beneficial to see how the performance changes with regard to

number of records kept. A Receiving Operating Characteristics curve (ROC) is commonly used to compare true positive rates and false positive rates for different thresholds [24]. A visualisation similar to the ROC-curve is a graph displaying f-score on the y-axis and fraction of records kept on the x-axis and how it changes for different thresholds. In a realistic setting, there is a compromise between performance and how many records to classify, in which this graph is intuitive. Normally, the more confident the model is, fewer records will be kept and the performance higher. An example of this graph can be seen in figure 3.

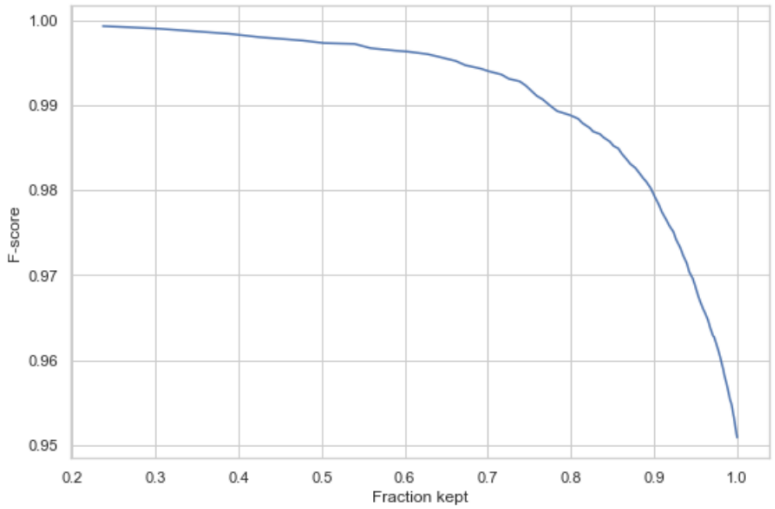

Figure 3: F-score and fraction kept of records for different thresholds. The higher the performance (f-score) the fewer records are classified.

2.6. Related work

To the best knowledge of the author, there are no studies that are directly related to the usage of metadata when classifying language in music. However, language identification in itself is a large field with many studies both for audio-based and text-based approaches. Also, metadata has previously been used extensively, both for classification and recommendation tasks, with some of them being related to genre and mood detection in music. As this report explores metadata in relation to classification of music, it relates to these topics and contributes to the gap where this specific area have not yet been investigated.

2.6.1. Classification using metadata

Slaney [25] argues that in multimedia information systems, using metadata about the content can often be more useful than the content itself. The author points to several examples within music similarity, movie recommendations and image tagging where the metadata have proved to be more effective.

Several studies have been conducted to classify genres based on extracting features from lyrics to build models [26, 27]. Çoban [28] investigated the effect of using audio and textual features when classifying Turkish music genres. The textual features were extracted from lyrics and represented in models such as word2vec. The audio was found to outperform the textual features, but the best results were obtained combining both features . Another study that classified genres from lyrics also included cultural and symbolic features extracted from web searches and listener tags. The authors found that the features extracted from lyrics were less effective while the cultural features related to the songs performed particularly well when classifying genres [29].

Classification of images have gained a lot of traction in recent time. While the content itself often is analysed through more advanced methods detecting colors, textures and shapes, J.Boutella and J.Luo [30] instead used metadata in digital image files for photo classification. With features like time stamps, presence of flash, subject distance, exposure time, and aperture value they managed to accurately discriminate indoor from outdoor pictures, sunset from non-sunset, and man-made from natural scenes. Other areas where metadata has been used is mood classification in music [31] and document classification in enterprises [32].

2.6.2. Recommendations based on metadata

Several studies have been made to build recommendation systems for music, both based on different sources of metadata and audio input [33, 34]. Wang et al. [35] used both metadata and contextual preferences to recommend music in

accordance with the user's preferences. The authors did this by learning embeddings of music pieces as feature vectors in a low-dimension continuous space. This was combined with global and contextual features for each user to build a model that recommend new songs to the user.

Knees and Schedl [36] explored different methods for music similarity estimation and music recommendation based on music context data. While they found that contextual data proved to have great potential for these use cases, most of the features shared similar shortcomings. Disproportionately more data were available for popular artists while data sparsity was common for less frequently played tracks. Noisiness in data sets and the cold start problem where data was not available for new tracks were two additional challenges.

2.6.3. Language classification

When studying language classification in text, Balwin and Lui [37] argues that there is a widespread misconception of language identification being a solved task. The authors point to isolated experiments based on homogeneous data sets with a small set of languages resulting in inflated confidence in proposed models, and that there is still a long way to go with regard to classification under realistic conditions.

Numerous studies have been made with regard to spoken language identification and sung language identification has also gained more traction the past few years. Similar techniques in spoken language identification have often been used in sung language identification, with the additional challenge that sung language is convoluted with instrumental music [38]. In addition to background music, the large variations found in singing voices makes it more difficult to implement language classifiers to music [8]. A commonly used technique is to convert the digital waveforms into spectrum-based feature vectors, to then use sophisticated models for classification [8, 9]. Convolutional neural networks have previously been used with promising results [39, 40].

At Spotify, Simon Durand (2018) built a classifier for 21 languages using only input from audio. The vocals were extracted from the song with a separation model before converted to a mel-spectrogram that was normalised. A convolutional neural network was used to output the language identification function. The mean activation per language was computed and language weighting applied to output the language confidence vectors of the track. The results were promising with an f-score of 0.88 for the whole set with additional performance improvements when only the more confident estimates were kept.

3. Method

3.1. Tools

In the making of this report, all code written was done in Python using Jupyter Notebook [41] and BigQuery was used to fetch data and to do parts of the pre-processing [42]. In Python a number of libraries were used. For data handling and pre-processing Pandas [43] and Numpy [44] were mainly used. For modelling, different classifiers from Sklearn were used in combination with evaluation methods from the same library [15].

3.2. Evaluating classifiers

Initial tests with various classifiers were made to get a rough estimate of performance and execution time. Random forest had good performance in the initial tests and was relatively quick compared to other models with an execution time of roughly an hour on the whole dataset. Random forest was therefore used to evaluate relevance and performance of features and combinations of features. After pre-processing the selected features, the following approach was taken:

1. Evaluate the model per feature.
2. Evaluate the model for combinations of features.
3. Evaluate different models on all features.
4. Use different confidence thresholds for predictions and evaluate the results.
5. Investigate robustness of the model when adding languages to the test set that the model was not trained on.
6. Compare model to another classifier based on audio.

To evaluate the different models, precision, recall, f-scores and accuracy were used throughout the report. Where performance of the whole set was reported without regard to the different languages, micro averages were used. In some cases, individual scores for different languages were analysed and confusion matrix was used to observe the different kinds of errors of the model. In addition to this, manual inspection of samples was made to observe the nature of cases

where the model and label did not match. When a threshold was used, the fraction of tracks kept and f-score were used to illustrate the relationship between performance and coverage of the model.

3.3. Limitations

Due to computational limitation, only a few different classifiers with limited variations of parameters were tested. In an ideal case, more classifiers would be tried with additional parameter tuning for all different cases. While there are tracks in over 50 languages at Spotify, a limitation was made to only study nine languages to achieve shorter execution time and easier validation. The labels in the training set were discovered to contain some mislabeled tracks. To what extent the data set was mislabeled was out of scope for this report to fully investigate, and the performance of the model and accuracy of results could have been affected by this.

4. Data and pre-processing

In this section, the data sources are presented along with the process of selecting features and languages. The pre-processing of all individual features are then described, followed by the methodology regarding missing data.

4.1. Data set

The data set that was used for training and testing consisted of 180,000 tracks in nine languages, with 20,000 tracks for each language. The language tags in this set comes from the record labels which have been providing tags to a part of the Spotify catalogue. The language tags for the tracks and all the features exist in various tables in BigQuery, with sizes varying from a couple of megabytes to around 50 terabytes.

4.2. Selection of Languages

When it comes to selection of languages to use a few criteria were desired to be fulfilled. To make better predictions, languages were chosen so that the classes contained a larger set of records and had fairly balanced classes [45]. There would not be full coverage of features no matter what languages were chosen, but some had better coverage than others. Countries where Spotify have recently launched or prepare to launch had worse coverage of features than those where Spotify have existed longer. For example, Hindi (hi) was excluded as Spotify had not released in India and had a relative small listening base and coverage of features. Empirically the labels for English (en) were suspected to be unreliable by people from Spotify and to contain songs that were in other languages, and was therefore excluded. The languages were also chosen on basis where some of them could be validated manually through knowledge of the team at Spotify. For this report nine languages with 20,000 samples in each were selected using Pandas sample() function, resulting in a data set of 180,000 songs. The selected languages were German (de), Spanish (es), Finnish (fi), French (fr), Italian (it), Japanese (ja), Dutch (nl), Portuguese (pt), Swedish (sv).

4.3. Selection of features

When it comes to feature selection there were a few different things to take into account. Aryes [46] describes how predictive modelling isn't a substitution for intuition but rather a complement, and that the same principle can be applied to obtaining relevant data and selecting features. Expert knowledge and the context of the problem plays a large part in the decision process in what data to use and what features to choose [12]. The features were chosen from the availability in the databases and the intuitive relevance of features. Two potential features were excluded: lyrics and artist name. While lyrics could be a very strong indicator to what language a song is, there was very limited availability of lyrics which led to it being excluded. Artist name could be a good indicator of language, but more advanced methods were needed to process it to a relevant feature, which in the end excluded it from this report.

Of the seven final features that were chosen, two were based on text analysis: song name and album name. These two features were used by a separate text classifier to create the features used in the models. Three of the features were based on numerical vectors that describes users, playlists and tracks. The last two features described the genres and regional popularity for each track. The selected features were multidimensional, which made the range of the feature space relatively large. The features representing tracks, playlists and users had 40 dimensions each. For song name and album name, there were 164 features each which was one for each language detected from the text-classifier. Region contained 176 dimensions, one for each country, and genre 2445. This resulted in a total of 3069 features.

4.4. Song and album name

The procedure and processing for the features song name and album name were identical and used the text classifier DetectLanguage. DetectLanguage is a simple API that works with Python and takes a string as input and outputs the language and confidence of the classification. The confidence score ranges

between 0 and 20 where higher is more confident. At the time of the report it supported 164 languages [47].

Song name	Album name
Ruokaa suussa	Putte Possun huviretki
Sixteen - 2017 New Vocal Version	Sixteen
Graceful days - 2017 New Vocal Version	Graceful days
So mach ich es	So mach ich es
One Love - 2012.06.22 @ NIPPON BUDOKAN	One Love
Benny Goodman No Choro	Uma História do Choro
Casa De Fados	Fados by Carlos Saura

Table 3: Example of song and album names of tracks.

Some words were quite general and did not give much information of the language of the song and could be misleading for the classifier. These words were identified as: *original, live, edit, remix, remake, acoustic, remastered, mix, DJ, version*. These words were removed from the string for both song name and album name. When these words were a subset of a longer word, they were not removed. For example *remixad* can be the Swedish conjugation of *remix*, which is still relevant and was not removed.

Song name	Language	Confidence
A Ti Jehova	cs	8
Pippi firar födelsedag - Del 4	sv	9.14
In der Geisterbahn	de	5.92
Helga-neidin Ei Pidä Enää Mennä Kylpyyn -	fi	7.91
Mar de Canal	en	7.66667
Rosi aus Tirol (feat. Geilomat)	de	0.01
Pusulan metsät	st	0.01
Hurry Xmas Originally Performed By Larc En Ciel	en	7.68

Table 4: Detected language with confidence scores from DetectLanguage for eight tracks.

As table 5 shows, the text classifier considered a large part of the tracks as English, although none of those tracks are labeled as English. Ideally, the language detected from DetectLanguage should match the label of that track,

but as we can see in the distribution that is not the case. The classifier might have a bias towards English, or songs and albums have titles in another language than the track actually is.

Language	Percent classified
en	20.2%
es	10.0%
de	9.2%
pt	8.7%
fr	7.8%
nl	7.8%
fi	7.2%
sv	6.8%
it	6.2%
ja	3.6%

Table 5: Detected languages classifications of song names in percent for the whole dataset.

Each language from DetectLanguage was added as a feature for the data set. Detect language have 164 possible languages, so 164 features were added each for song name and album name. The model gave output for the most confident language for each track, resulting in the features to be very sparse with one value per row. Where the values were missing, 0 were imputed as the absence of values can be interpreted as zero confidence. A sample of the created features can be seen in table 6.

Song name	t_aa	t_en	t_fi	...	t_sv	t_ta	t_yo
En sång om sorg och glädje	0	0	0	...	12.7	0	0
Better now	0	10.0	0	...	0	0	0
Samanlainen onni	0	0	10.84	...	0	0	0
Crapuleux	0	0	0	...	0	8.19	0
Kejime	0	0	0	...	0	0	0.01

Table 6: Sample of song name features.

4.5. Track vectors

The track vectors describing the characteristics of a track were trained directly at Spotify by a word2vec model. While word2vec models are originally used for

documents with text, they were in this case trained with tracks and playlists in mind. Each playlist was treated as a "document" with each track within that playlist as a "word". The surrounding tracks were considered as the context for training the vectors. Vectors were then created to minimize the distance between tracks that share similar context. These track vectors reside in BigQuery and were fetched to be pre-processed before added to the model. The process of converting each element of the vector into features can be seen in table 7 and 8.

id	Track vectors
0	[0.07, 2.10, -0.94, ... , 0.07, 2.10, -0.94]
1	[-0.42, -0.29, -0.54, ... , -0.52, 1.10, 0.24]
2	[-0.50, -0.41, -0.41, ... , 0.29, 1.43, -0.44]

Table 7: One 40-dimensional track vector for each track id.

id	vector_1	vector_2	vector_3	...	vector_38	vector_39	vector_40
0	0.07	2.10	-0.94	...	0.07	2.10	-0.94
1	-0.42	-0.29	-0.54	...	-0.52	1.10	0.24
2	-0.50	-0.41	-0.41	...	0.29	1.43	-0.44

Table 8: 40-dimensional track vectors converted to 40 features.

4.6. Playlist and user vectors

Playlist and user vectors were based on the track vectors and were generated at Spotify prior to this thesis and were available in BigQuery. As the playlist and user vectors represent a specific user and playlist, average vectors were needed to be calculated for each individual track to be used as features.

User vectors:

The user vectors were produced by taking a weighted average of the track vectors in each user's listening history. The weights assigned depends on the listening context for each tracks plays. A track that was saved by a user was for example weighted in a different way than a track that was skipped within 30 seconds. There was also a decay applied so that older listens count less towards the user

vector. To generate a user vector for a specific track, an average user vector for that track is calculated to be used as a feature. The process to generate the average user vectors for each track was the following:

1. Get the user ID for everyone that played the track for more than 30 seconds the past six months.
2. Fetch the user vectors for those users.
3. Take the average of the user vectors for each track.

Playlist vectors:

Similar to user vectors, a weighted average was calculated for each playlist based on the track vectors. To go from the playlist vectors in BigQuery to an average playlist vector for each track, the following steps were taken:

1. Get the IDs of all the playlists that the track is in.
2. Fetch the playlist vectors for those playlists.
3. Take the average of the playlist vectors for each track.

Processing of playlist and user vectors:

The processing of the data in BigQuery to produce the average playlist and user vectors was quite heavy as it needed to look through all created playlists and all users who had listened to a track. The process for playlist vectors is shown in table 9 to 11 where *id* represents the id for each track. The averages of the vectors and expansion to columns for separate features was done in Pandas in Python. The process below was also made for user vectors.

id	Playlist vectors
0	[0.20, 2.20, -1.42, ... , 0.52, 1.12, -1.37]
	[-0.87, -0.23, -0.76, ... , -0.34, 1.73, 0.13]
	[-0.51, -0.63, -0.99, ... , 0.14, 1.82, -0.99]
	[0.12, 2.55, -1.12, ... , 0.66, 1.52, -0.22]
1	[-0.76, -0.87, -0.32, ... , -0.32, 1.86, 0.41]
	[-0.92, -0.60, -0.76, ... , 0.78, 1.91, -0.11]

Table 9: Multiple 40-dimensional playlist vectors for each track id.

id	Average playlist vectors
0	[0.10, 2.00, -1.24, ... , 0.23, 1.20, -1.32]
1	[-0.13, -0.91, -0.21, ... , -0.33, 1.56, 0.10]
2	[-0.92, -0.61, -0.96, ... , 0.14, 1.62, -0.42]

Table 10: 40-dimensional average playlist vectors, one for each track id.

id	vector_1	vector_2	vector_3	...	vector_38	vector_39	vector_40
0	0.10	2.00	-1.24	...	0.23	1.20	-1.32
1	-0.13	-0.91	-0.21	...	-0.33	1.56	0.10
2	-0.92	-0.61	-0.96	...	0.14	1.62	-0.42

Table 11: 40-dimensional average playlist vectors as features.

4.7. Region

Regional popularity from 176 countries were collected for all the tracks that it was present for. The popularity scores were normalized per country (region) to be between 0 and 1, where a higher score means that the track has been more frequently played there. There was only a score if the track had above a certain threshold of plays in a specific region. This meant that some tracks had scores for 15 regions while others had for only one. For the popularity scores that were missing from tracks zeroes were imputed to indicate no popularity. For some of the country codes, there were duplicate entities where one was lower-case and one upper-case which had to be merged together. One feature was created for each region and the popularity for each track and region were inserted, as can be seen in table 13.

id	Region	Popularity
0	[DK, US, CH, AT, NL]	[0.010, 0.140, 0.621, 0.132, 0.012]
1	[PA, MX, NZ]	[0.020, 0.006, 0.132]
2	[DE, RO, NI, TH, LU, HK, ...]	[0.225, 0.011, 1.02e-07, 0.011, ...]
3	[BE, NL]	[0.064, 8.58e-05]
4	[CA, SE, ES, NL, FR, FI, NO]	[2.26e-08, 1.30e-07, 0.042, 0.014, ...]

Table 12: Regions and corresponding popularities for each track id.

id	DK	DE	PA	...	US	ES	NL
0	0.010	0	0	...	0	0	0.012
1	0	0	0.020	...	0	0	0
2	0.225	0	0	...	0	0	0
3	0	0	0	...	0	0	8.58e-05
4	0	0	0	...	0	0.042	0.014

Table 13: 176-dimensional regional popularity features for each track id.

4.8. Genre

There were many different genres among the tracks in Spotify's database and for this data set 2445 genres were present. The genres for each track were defined as audio distance and the closer the distance to a genre, the more that genre represented the track. Each track could have audio distance for zero or more genres represented. The genres in the original data only had a score if they were under a certain threshold, otherwise they were missing. This resulted in a very sparse matrix where each track had a clear majority of missing values. In a similar way as the features for regional popularity were pre-processed, each genre was represented as a feature with the audio distance as values. If a value was missing, an arbitrary high value of 10 was imputed in its place to indicate a long distance from that genre. Examples of genres and audio distance, and transformation into features can be seen in table 14 and 15.

id	Genres	Audio distance
0	[italian pop, mexican pop]	[2.14, 2.56]
1	[belgian hip hop]	[4.70]
2	[grupera, regional mexican, banda, cumbia, ...]	[3.31, 3.96, 3.99, 4.21, ...]
3	[kleine hoerspiel, hoerspiel]	[2.96, 2.98]
4	[swedish jazz, classic swedish pop, swedish folk, ...]	[2.54, 2.06, 2.29, ...]

Table 14: Example of genres and audio distance for each track id. In the first row, italian pop had the audio distance of 2.14 and mexican pop 2.56.

id	italian pop	belgian hip hop	...	swedish jazz	hoerspiel
0	2.14	10	...	10	10
1	10	4.70	...	10	10
2	10	10	...	10	10
3	10	10	...	10	2.98
4	10	10	...	2.54	10

Table 15: The genres transformed into 2445 separate features.

4.9. Missing data

A challenge was that lots of data were missing for many tracks. While there are different ways to handle this problem, simple imputations were chosen as the final method. To be able to simulate a real-case scenario where there in many cases will be missing data, rows were not removed, but instead imputed to keep the whole data set. All the features were multidimensional and had some missing values in many of the features.

Different approaches of imputing were made for different features. For song name, album name and region, a static value of zero was imputed to indicate an absence of a signal. For example, in regional feature *sv* (Sweden), an imputed 0 indicates zero popularity in Sweden. The genre features were measured in audio distance between the track and genre, meaning a low value having a high probability to belong to that genre. A track with missing value for the genre feature *reggae* was imputed with a high static value to be interpreted as far away from that genre. For the track vectors, playlist vectors and user vectors, mean vectors were calculated on the training set. A mean vector across all records in the training set were imputed on tracks that had missing values, both in the training and test sets. While there are more advanced methods of imputation, the limited computational power available and the high number of features in a relatively large data set resulted in more simple imputations being used.

The coverage of the different features can be seen in figure 4. Song name and album name were present in 100% of the tracks, as each track had a song name

and belonged to an album. Other features, such as track vectors, had 48% coverage.

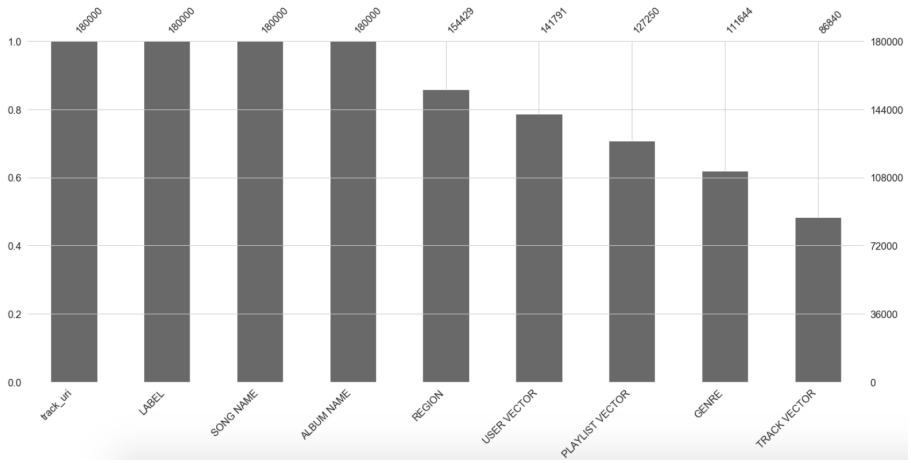

Figure 4: Fraction of coverage for each feature in the labeled data set.

5. Results

The results section consists of seven parts. In 5.1 the performance of each individual feature was measured. In section 5.2 various combinations of features were tested and the results evaluated. Section 5.3 display the results when all available features were used in different types of classifiers. In 5.4, the confidence probabilities of the model were used instead of discrete classes, and various threshold of this confidence were investigated to see how it affected performance. 5.5 contains results of manual inspection of cases where the classifier was wrong, and section 5.6 looks at how the classifier behaved if additional languages were added to the test set. Finally, section 5.7 compares the final model to an existing classifier that only uses audio as input signal.

5.1. Individual features

A random forest model with 150 trees, GINI-index as criterion and automatic max-depth was used for all features separately. For each feature, a subset was used where the feature had full coverage. The size of these subsets varied from feature to feature. 20% of the data was used as a hold-out set to evaluate the classifier. In table 16 each feature's coverage of the full set is shown together with f-score (micro) and accuracy.

Feature	*Accuracy*	*F-score*	*Feature coverage*
Song name	71.9%	0.72	100%
Album name	68.2%	0.68	100%
Track vectors	97.4%	0.97	48%
Playlist vectors	95.1%	0.95	71%
User vectors	94.2%	0.94	79%
Region	87.7%	0.88	86%
Genre	95.2%	0.95	62%

Table 16: Summary with performance metrics and feature coverage of all features for a random forest model with 150 trees. Album name had relatively low performance with 100% coverage of the features in the data set, while track vectors had high performance with only 48% coverage in the data set.

5.2. Combinations of features

The same model was used for all combination of features for the whole data set to find best combination of features. In each case all the data was used, no matter how many missing values there were. A "base-case" was used for all combinations where the features for song name and album name were present. There was a 100% coverage of those features and in addition all combinations of the other features were tested with a random forest classifier with 150 trees and imputation with mean.

The results can be seen in table 17 where the best performing selection was when all features except user vectors was used, with an f-score of 0.953 and accuracy of 95.4%. The second-best result was with all features and only slightly below the top one with f-score of 0.954 and accuracy of 95.3%.

Combination	*Accuracy*	*F-score*
Base + (Playlist v, Track v, Region, Genre)	95.4%	0.954
Base + (Playlist v, Track v, User v, Region, Genre)	95.3%	0.954
Base + (Track v, Region, Genre)	95.2%	0.952
Base + (Playlist v, User v, Region, Genre)	95.2%	0.952
Base + (Playlist v, Region, Genre)	95.2%	0.952
Base + (Track v, User v, Region, Genre)	95.2%	0.952
Base + (Playlist v, Track v, User v, Genre)	95.1%	0.951
Base + (Playlist v, User v, Genre)	95.1%	0.951
Base + (Track v, User v, Genre)	95.0%	0.950
Base + (User v, Region, Genre)	94.9%	0.949
.	.	.
.	.	.
Base	81.7%	0.819

Table 17: Performance on combination of features with a random forest model with 150 trees. Base represents the features song name and album name. In addition to base, all other combinations of features were tried.

5.3. Evaluating different classifiers

After the initial tests with individual features and combination of features, various models from Sklearn with mainly default settings were used on the whole data set with results shown in table 18. Mean imputation were used to impute

missing values and a 20% hold-out set was used to test the models. The random forest model that previously was used for individual features and combination of features also performed best when compared to other classifiers.

Model	*Accuracy*	*F-score*	*Description*
Random forest	95.4%	0.952	Default values and 150 trees
XGBoost	94.4%	0.944	Default values
KNN	93.2%	0.932	Default values and K = 10
Logistic regression	92.8%	0.929	Default values

Table 18: Performance of various models on the whole data set with mean imputations for missing values.

For the best performing model, random forest with 150 trees, performance for individual languages and confusion matrix is displayed in table 19 and figure 5.

Language	*Precision*	*Recall*	*F-score*	*Support*
de	0.95	0.95	0.95	3993
es	0.94	0.95	0.95	3966
fi	0.99	0.98	0.98	4122
fr	0.94	0.92	0.93	3944
it	0.93	0.92	0.93	3971
ja	0.89	0.94	0.92	4045
nl	0.97	0.97	0.97	3978
pt	0.97	0.96	0.97	3971
sv	0.97	0.95	0.96	4010

Table 19: Performance for individual languages with a random forest classifier with 150 trees for the whole data set. Support indicates the number of occurrences of each class in the test set.

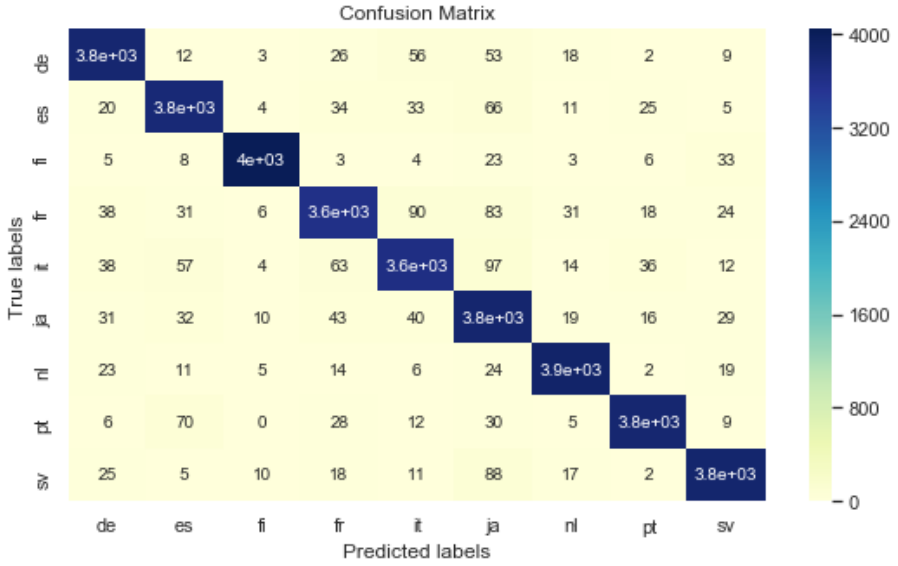

Figure 5: Confusion matrix for individual languages with a random forest classifier with 150 trees for the whole data set.

5.4. Threshold for confidence

Using the classifier based on the random forest model with 150 trees, the probabilities were used instead of the discrete classifications for all features and the whole data set. Different thresholds were used for these probabilities to investigate how the model performed when it was more or less confident in the predictions. In table 20 fraction of songs kept, accuracy and f-score are displayed for different thresholds. For example, with a probability threshold over 0.50, 92.84% of the tracks were kept and classified with an accuracy of 97.41%. In figure 6, f-score is plotted over the fraction of songs kept for certain thresholds.

Threshold	Fraction kept	Accuracy	F-score
0.10	100%	95.2%	0.952
0.20	99.9%	95.2%	0.952
0.30	98.6%	95.8%	0.958
0.40	96.3%	96.5%	0.965
0.50	92.8%	97.4%	0.974
0.60	88.3%	98.2%	0.982
0.70	82.7%	98.7%	0.987
0.80	75.0%	99.2%	0.992
0.85	69.4%	99.4%	0.994
0.90	61.5%	99.6%	0.996
1.0	12.6%	100%	1.000

Table 20: Fraction kept of songs, accuracy and f-score are shown for different threshold of confidence of the model. With a threshold of 0.50, 92.8% of the tracks are classified with an accuracy of 97.41%.

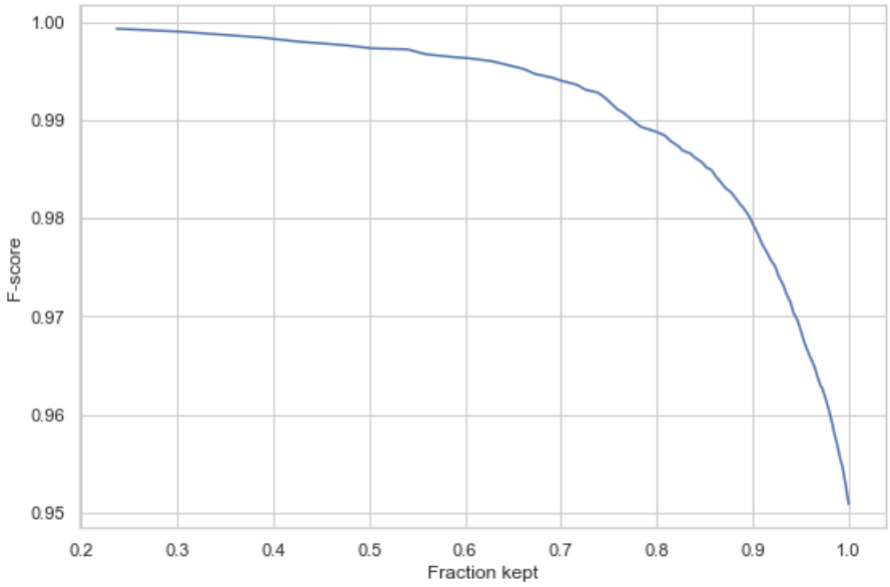

Figure 6: F-score plotted over the fraction of songs kept for certain probability thresholds of the model. For 90% of tracks to be classified, an f-score of around 0.98 would be achieved.

Where to put the threshold is dependent on the context of the situation where the classifier is implemented as it is a compromise between performance and fraction of the songs to classify. An example where the threshold was put at 0.85 where 69.4% of the most confident predictions were kept was investigated

further. The average across all languages was an f-score of 0.994 and accuracy of 99.4% with results for individual languages in table 21. Figure 7 shows the confusion matrix with the same threshold.

Language	*Precision*	*Recall*	*F-score*	*Support*
de	1.00	1.00	1.00	2535
es	0.99	0.99	0.99	2636
fi	1.00	1.00	1.00	3569
fr	1.00	0.99	0.99	2148
it	0.99	0.98	0.99	2032
ja	0.99	0.99	0.99	2557
nl	1.00	1.00	1.00	3144
pt	0.99	1.00	0.99	3030
sv	1.00	0.99	0.99	3236

Table 21: Performance for individual languages with a threshold at 0.85, keeping 69.4% of the most confident predictions.

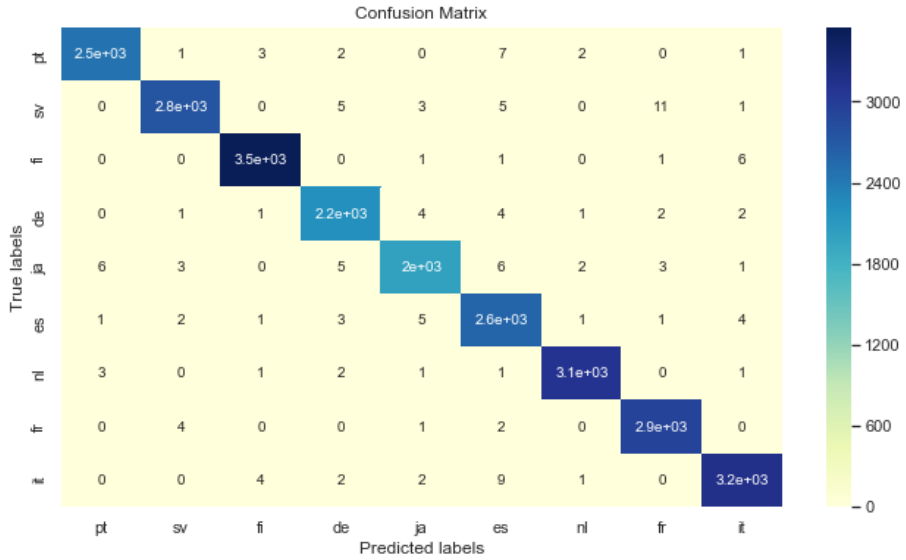

Figure 7: Confusion matrix the languages with a threshold of 0.85 where 69.4% of the most confident predictions were kept.

5.5. Validation of mismatches

While investigating all classifications manually was unfeasible, manual inspections were made for a subset of the languages; sv (Swedish), fr (French), fi

(Finnish) and es (Spanish). The inspection was made using a confidence threshold of 0.85 keeping 69.4% of the most confident predictions and looking at the tracks where the classifier and labels did not match. False positives (FP) and false negatives (FN) from the model were inspected manually to see the actual number of FP and FN. For both false positive and false negative, the classifier was correct and labels wrong in about 50% of the cases, with similar error rates across all investigated languages. More details can be seen in table 22.

Language	*FP*	*Actual FP*	*FN*	*Actual FN*
sv	16	7	18	9
fr	19	9	15	12
fi	10	7	9	2
es	11	6	25	12

Table 22: Manual validation of false positives and false negatives for languages sv, fr, fi and es. The classifications were inspected from a random forest model with a threshold of 0.85, keeping 69.4% of the most confident predictions.

5.6. Reliability with additional languages

In a real-world scenario, the languages of the test set might be different from the languages the model is trained on. To emulate this and see the classifiers resilience to input with wrong classes, a new test set was created containing additional languages. The model was trained on a reduced set of 90,000 tracks with the nine original languages (de, es, fi, fr, it, ja, nl, pt, sv). The test set consisted of around 1000 tracks each of the original nine languages plus 10 additional languages: Croatian (hr), English (en), Hindi (hi), Afrikaans (af), Thai (th), Malaysian (ml), Punjabi (pa), Hungarian (hu), Danish (da), Kannada (kn). The distribution of languages for the new test set can be seen in figure 8.

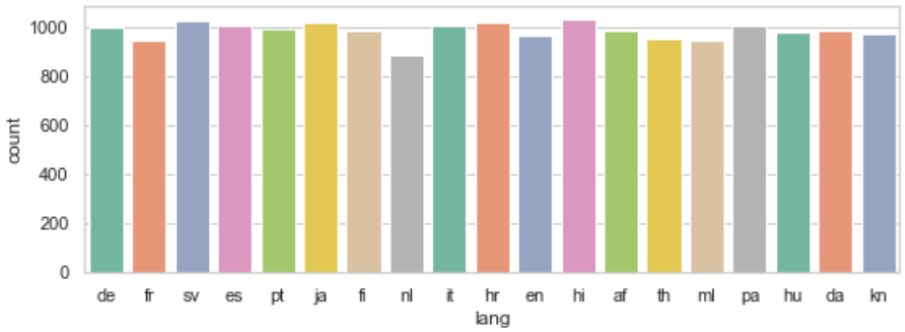

Figure 8: Distribution of languages for a new test set containing 19 languages, where the nine on the left side (de, fr, sv, es, pt, ja, fi, nl, it) belong to the training set.

With a random forest classifier with 150 trees and a threshold of confidence at 0.85, table 23 displays the distribution of the languages kept after classification. 49.0% (4362 of 8893) of the correct nine languages were kept and 0.23% (23 of 9876) from the additional languages remained. In an ideal situation only the tracks from the nine original languages would be kept as those are the only labels the model is trained on. The rest should be filtered away by the model. The languages that remained from the additional 10 languages were 11 in Danish (da), 9 in English (en) and 3 in Afrikaans (af).

Language	Remaining	Original count	Percent remaining	Language in training set
nl	700	860	81%	Yes
fi	689	939	73%	Yes
pt	636	995	64%	Yes
es	479	1001	48%	Yes
de	472	1000	47%	Yes
sv	415	1012	41%	Yes
fr	389	930	42%	Yes
ja	366	1021	36%	Yes
it	306	998	31%	Yes
da	11	988	1.1%	No
en	7	961	0.7%	No
af	1	991	0.1%	No
hr	0	1009	0%	No
hi	0	1029	0%	No
th	0	961	0%	No
ml	0	955	0%	No
pa	0	1002	0%	No
hu	0	990	0%	No
kn	0	978	0%	No

Table 23: Number of tracks remaining for each language for the data set with 19 languages after entering a threshold of 0.85. Ideally, only the languages that exist in the training set should remain.

After manual inspection, the labels for many of the tracks that remained in the additional 10 languages were incorrect and the classifier correct. 7 of 7 tracks in English (en) were correctly classified by the model, but had incorrect ground truth of the label, leading to false positive when it in fact was true positive. All 11 tracks in Danish (da) that remained were classified as Finnish which also was true positive. The song in Afrikaans (af) was predicted as Dutch (nl) by the model but was actually in Flemish which is similar to Dutch, so neither the ground truth or classifier were correct. After those corrections, all the tracks from the additional languages were removed except for the track in Afrikaans. For the original languages, the result of the classification can be seen in table 24.

Language	*Precision*	*Recall*	*F-score*	*Support*
de	1.00	1.00	1.00	472
es	1.00	1.00	1.00	479
fi	1.00	1.00	1.00	689
fr	1.00	1.00	1.00	389
it	1.00	0.99	1.00	306
ja	1.00	1.00	1.00	366
nl	1.00	1.00	1.00	700
pt	1.00	1.00	1.00	636
sv	1.00	1.00	1.00	415

Table 24: Performance for each language with around 1000 tracks in 19 languages and a threshold of 0.85 with 49% of the most confident prediction kept.

5.7. Comparison with audio classifier

The final model using random forest with 150 trees was compared to another model based solely on the audio input created by Durand (2018) at Spotify. The audio classifier extracted the vocals from the songs and used a convolutional neural network to train the model. Both models were trained and tested on the same set of 180,000 tracks that were used in previous chapters. A random sample of 34,888 tracks (around 20%) was used as a hold-out set to test on, while the rest was trained on. Figure 9 shows the graph for micro f-score and fraction kept for both models for different threshold. In figure 10 the same data is shown, but only more confident predictions are kept resulting in higher f-scores and fewer tracks classified.

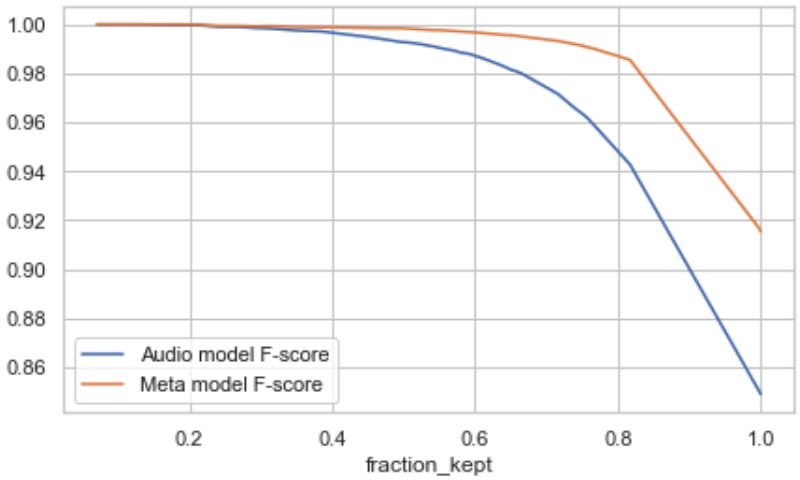

Figure 9: Comparing f-score and fraction of tracks classified for different thresholds of the audio model and metadata model. F-score is increasing for both classifiers as fraction of tracks classified is decreased.

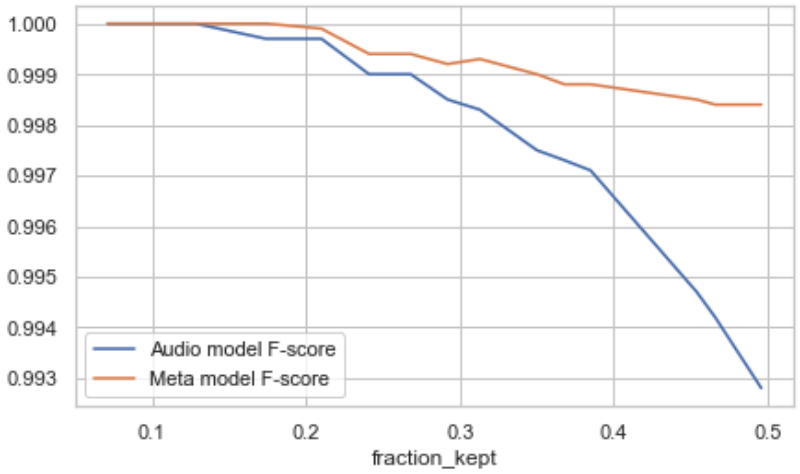

Figure 10: Comparing f-score and fraction of tracks classified for different thresholds of the audio model and metadata model. Showing the same data as figure 9 but where fraction kept for the models is less than 50%.

After selecting thresholds for the two models so that 50% of each model's most confident predictions were classified, the f-score was 0.998 for the metadata model and 0.993 for the audio model. When looking at the overlapping of the classified tracks both models kept, 64% was overlapping. The whole test set

consisted of 34,888 tracks where 23,724 (68%) were classified by either or both of the models, 17,444 (50%) by the models respectively and 11,164 (32%) by both models. These counts are displayed in figure 11.

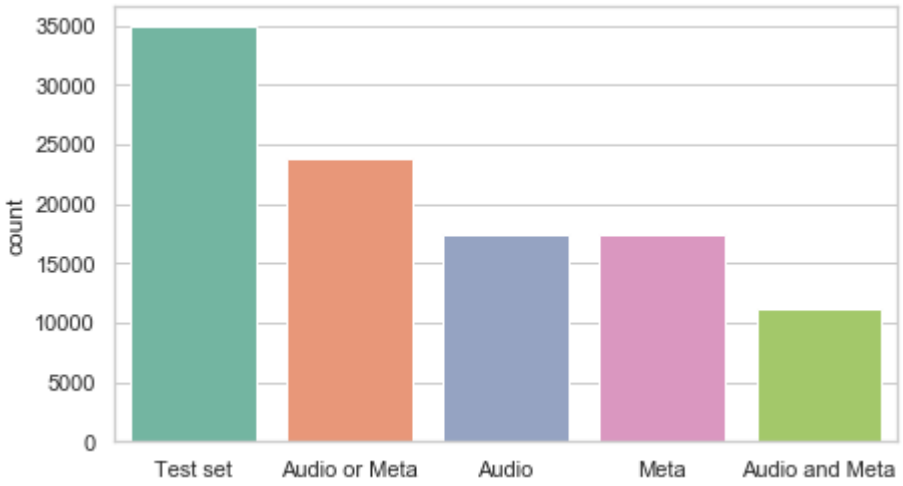

Figure 11: Number of classifications for metadata model and audio model when keeping the 50% most confident predictions of the test set for each model.

6. Discussion

6.1. Feature relevance

The relevance of features and how they performed in combination with each other were investigated with an identical set up for a classifier. Individually, the features album name and song name performed worst with f-scores of 0.68 and 0.72 respectively, but had the benefit of presence in all the tracks used. Track vectors on the other hand performed best individually with an f-score of 0.97 but also the worst coverage with 48% of the tracks. Since the user and playlist vectors themselves were based on a weighted average, the method of averaging them again to create the final feature in this report could be questioned. They did however prove to be performing well and had relatively high coverage. As all the features were different in number of dimensions, how imputations are made within these dimensions and the characteristics, using the same model for each individual test could potentially lead to misleading results. It did however give an indication for the relevance of the different features. When the different combinations of features were studied, the top results were fairly similar with an f-score of $\pm$ 0.002. While all features were used in the final model, one could argue that a simpler model with fewer features would be to prefer to reduce complexity as the performance was similar. The fact that track vectors, playlist vectors and user vectors performed well was an indication that both playlists and users at Spotify are fairly language consistent, which were reflected in those vectors.

With the large amount of missing values, a preferred method would be to exclude tracks with missing features. As the report aimed to simulate a realistic case where as many tracks as possible are desired to be classified, removing those was not an option. This problem was solved with a basic mean imputation, but could possible benefit from more advanced methods or from Spotify's side to expand the collection of metadata to increase coverage.

6.2. Model performance

Due to a large set of features and a limited amount of computational power, there were limitations to how many models with different parameters that could be tried in combination with different features. A random forest with 150 trees proved to perform the best with a micro f-score of 0.952 for the whole set. XGboost had the potential benefit of being able to handle missing values without imputations in the pre-processing phase, but ended up performing better when static values were imputed. Depending on the aim when implementing the classifier, different thresholds could be used based on a compromise between performance and fraction of tracks classified. Figure 6 displays how the f-score is increasing with a higher threshold and decreasing the number of classified tracks. The performance across different languages was fairly consistent when using the final model, but less consistent for the individual features (see tables 25 to 31 in the appendix). In a few cases, es (Spanish) was mislabeled as (Italian) and the other way around, which could make sense due to the proximity in the languages. In another case, a few French tracks were classified as Swedish. An example of this was the French song "Étienne" by the locally famous artist Christer Björkman who is mainly listened to by Swedes, which is likely to have made the classifier tag it as Swedish.

6.3. Label reliability

The ground truth in the data was discovered not to be completely reliable throughout the report. Due to the large quantity of tracks and limited scope of the thesis, a manual inspection of a larger sample to validate the ground truth was not possible. Instead the cases where the model with high confidence didn't match the labels were manually inspected for sv (Swedish), fr (French), fi (Finish) and es (Spanish). With regard to the False Positives and False Negatives of the classifier with a confidence threshold of 0.85, roughly 50% of the classifications were actually correct but mislabeled in the data set (see table 22), which could indicate higher performance than presented throughout the report. However, the fact that the investigated sample was very small and limited to a

number of languages, and that cases where both the classifier and ground truth was wrong were not included, one should be careful with generalizing these findings to the rest of the studied data set.

6.4. Robustness with input from other classes

After training on the selected nine languages, an additional 10 languages were added to the test set see how the classifier would behave in a realistic situation where the input of languages don't always match the languages the model was trained on. When a higher threshold was used only a few tracks from the untrained languages remained, which after manual inspection could be discarded as faults in the labels of the data. For lower thresholds, the explanation for remaining tracks in the added languages could often be found. For example, English songs by Swedish artists that are mainly played in Sweden would have user vectors, regional vectors and possibly other features pointing towards Swedish, leading to a Swedish classification. Norwegian and Danish tracks classified as Swedish could be based on similar errors.

6.5. Comparison with audio classifier

As a final step, the final model was compared with an existing audio classifier at Spotify. The classifiers were trained and tested on the same data and the results showed that the model based on metadata generally performed better than the one based on audio. However, due to the not completely reliable ground truth, one cannot be certain about the difference between the models, especially for higher f-scores which was previously investigated. Also, it is hard to draw general conclusions about the different performances of the two models when applied to other data sets and languages as this was done with limited dataset for a limited number of languages. When inspecting the overlapping of the classified tracks after using a threshold for both models, 64% were found to be classified by both models. While each model classified 50% for that threshold, the two models together classified around 70% of the tracks. This is an indication that the two models perform well on different kinds of tracks which could be useful when combining the two models in a separate classifier.

7. Future work

For future work, there are a number of areas that could be investigated for further improvements of the performance.

7.1. Deployment of classifier

By the end of the thesis, the final model was used to tag 26 million tracks in the Spotify catalogue. The classifier was trained using the final random forest classifier in the thesis with the same nine languages this report was based on. The output was confidence values for each language for each of the tracks, to enable filtering on different thresholds later when implementing the tags, depending on desired compromise between performance and coverage. With the current setup, additional languages can be added for the model to train on. The output could also potentially be used in combination with the audio classifier in a separate model.

7.2. Text classifier

The text features song name and album name had the benefit of coverage in all the tracks. However, the text classifier Detectlanguage proved not to be very accurate and had a bias towards English in the classification of song and album name, although none of the tracks were in English. While no other text classifier was tried during this thesis, additional improvements could probably be found using a more sophisticated text classifier.

7.3. More advanced imputations

A challenge throughout the report was the large fraction of missing data. In the report a combination of static values and mean values were used for imputations. There are more advanced methods of imputations available, which could be worth investigating to improve performance.

7.4. Train and test on additional languages

Due to a limited scope of the thesis, a subset of nine languages were studied in the report. It would be interesting to see how a larger set of languages would perform, and would also increases the value of implementing it in actual production.

7.5. Additional features and models

A set of nine features was used in the classifier in the report. Additional features that could be explored are lyrics and artist names. Lyrics was unavailable at the time of the thesis, but could potentially be used in another setting. For lyrics a text classifier could be implemented and as lyrics contains a large corpus of text, a classifier would probably have an easier time detecting language rather than the rather short title of song and album. For artists, rather than detecting the language of the individual artist, a classifier could be trained using the artist names as input. In the report a select few models were used, mainly with default settings. A natural next step from this report is to do explore the performance of additional models and to do further parameter tuning, both for individual features and for the whole set of features.

7.6. Combine audio model with metadata model

When using the most confident predictions for both the audio classifier and metadata classifier, it could be shown that they were both performing well and also not completely overlapping. This indicates that they may perform well on different kind of tracks and that a combination of the two classifiers could be useful. Either implementing the output of the audio classifier in the metadata model or using the predicted confidence values from both classifiers in a new separate model would be possible alternatives.

7.7. Reliability of ground truth

While many improvements could be made with regard to the modelling and features, a problem in this report was the suspected unreliability of the labels in

the data set. For future work and improvements of feature selection, imputation, models and parameter tuning, a more reliable ground truth to train and test on would be favourable.

8. Conclusion and summary

This thesis aimed to investigate how metadata from Spotify could be used to identify the language of songs in a dataset containing nine languages. Features based on song name, album name, genre, regional popularity and vectors describing songs, playlists and users were analysed individually and in combination with each other in different models.

Song name and album name resulted in the lowest performance individually based on the methods applied with an accuracy around 70%, but had the benefit of being present in all the tracks in the data set. When the features from the track vector were used, an accuracy of more than 97% was achieved, but only about half of the tracks in the dataset had that feature. For all features, a random forest classifier proved to be most effective with an accuracy of 95.4% for the whole data set. Performance was also investigated when the confidence of the model was taken into account, and when only keeping more confident predictions from the model, accuracy was higher. For example, when keeping the 70% most confident predictions in the test set, accuracy was 99.4%. The model was also demonstrated to be effective when additional languages not present in the training model were added to the test set. The final classifier proved to be performing better than an existing classifier based on audio input when trained and tested on the same data set. Future work could include a more extensive study of classifiers, data with more accurate ground truth, more advanced methods of imputation for missing data and combining the audio with the metadata classifier.

References

[1] J. Kim, C. Nam, M. H. Ryu, What do consumers prefer for music streaming services?: A comparative study between Korea and US, Telecommunications Policy 41 (4) (2017) pp. 263–272.

[2] S. K. Lee, Y. H. Cho, S. H. Kim, Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations, Information Sciences 180 (11) (2010) pp. 2142–2155.

[3] E. Zangerle, C.-M. Chen, M.-F. Tsai, Y.-H. Yang, Leveraging affective hashtags for ranking music recommendations, IEEE Transactions on Affective Computing (2018) pp. 1–2.

[4] S. Oramas, V. C. Ostuni, T. D. Noia, X. Serra, E. D. Sciascio, Sound and music recommendation with knowledge graphs, ACM Transactions on Intelligent Systems and Technology 8 (2) (2016) pp. 1–21.

[5] V. Krishna, Audience got furious and walked out of AR Rahman's concert for singing all tamil songs and not hindi, All India Roundup,URL `https://allindiaroundup.com/entertainment/audience-walked-out-of-ar-rahman-concert-as-he-sang-all-tamil/songs-instead-of-hindi/`.

[6] Spotify technology s.a. announces financial results for third quarter 2018, Press Release Details.
URL `https://investors.spotify.com`

[7] B. Roark, M. Saraclar, M. Collins, Discriminative n-gram language modeling, Computer Speech & Language 21 (2) (2007) pp. 373–392.

[8] W.-H. Tsai, H.-M. Wang, Automatic identification of the sung language in popular music recordings, Journal of New Music Research 36 (2) (2007) pp. 105–114.

[9] M. Mehrabani, J. H. Hansen, Language identification for singing, in: Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on, IEEE, (2011), pp. 4408–4411.

[10] C. M. Bishop, Pattern recognition and machine learning, Springer, (2006).

[11] M. Sokolova, A systematic analysis of performance measures for classification tasks, Information Processing Management 45 (4) (2009) pp. 427–437.

[12] K. Johnson, M. Kuhn, Applied Predictive Modelling, Springer, (2013).

[13] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning: Data mining, inference, and prediction. 2nd ed., corrected at 11th printing, Springer, (2017).

[14] Z. Abu-Aisheh, R. Raveaux, J.-Y. Ramel, Efficient k-nearest neighbors search in graph space, Pattern Recognition Letters (May) (2018) pp. 1–10.

[15] Scikit-learn, `https://scikit-learn.org/`.

[16] L. Rokach, O. Maimon, Top-down induction of decision trees classifiers - a survey, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 35 (4) (2005) pp. 476–487.

[17] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, Vol. 13-17, ACM, (2016), pp. 785–794.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, Curran Associates, Inc., (2013), pp. 3111–3119.

[19] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, Vol. abs/1301.3781, (2013).

[20] F. Vasile, E. Smirnova, A. Conneau, Meta-prod2vec: Product embeddings using side-information for recommendation, ACM, (2016), pp. 225–232.

[21] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, ACM, (2014), pp. 701–710.

[22] G. I. Webb, Encyclopedia of Machine Learning, Springer US, (2010).

[23] Y. Yang, An evaluation of statistical approaches to text categorization, Information Retrieval 1 (1) (1999) pp. 69–90.

[24] D. Powers, Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation, Journal of Machine Learning Technologies 2 (1) (2011) pp. 37–63.

[25] M. Slaney, Web-scale multimedia analysis: Does content matter?, IEEE MultiMedia 18 (2) (2011) pp. 12–15.

[26] R. Mayer, R. Neumayer, A. Rauber, Rhyme and style features for musical genre classification by song lyrics., in: Ismir, (2008), pp. 337–342.

[27] Z. Bodó, E. Szilágyi, Connecting the last.fm dataset to lyricwiki and musicbrainz. lyrics-based experiments in genre classification, Acta Universitatis Sapientiae, Informatica 10 (2) (2018) pp. 158–182.

[28] O. Çoban, Turkish music genre classification using audio and lyrics features, Süleyman Demirel University,Journal of Natural and Applied Sciences 21 (2) (2017) pp. 322–329.

[29] C. McKay, J. A. Burgoyne, J. Hockman, J. B. Smith, G. Vigliensoni, I. Fujinaga, Evaluating the genre classification performance of lyrical features relative to audio, symbolic and cultural features., in: ISMIR, (2010), pp. 213–218.

[30] M. Boutell, J. Luo, Beyond pixels: Exploiting camera metadata for photo classification, Pattern Recognition 38 (6) (2005) pp. 935–946.

[31] K. Trohidis, G. Tsoumakas, G. Kalliris, I. P. Vlahavas, Multi-label classification of music into emotions., in: ISMIR, Vol. 8, (2008), pp. 325–330.

[32] A. Karjalainen, T. Paivarinta, P. Tyrvainen, J. Rajala, Genre-based metadata for enterprise document management, Vol. 2, IEEE, (2000), pp. 1–2.

[33] A. Abdul, J. Chen, H.-Y. Liao, S.-H. Chang, An emotion-aware personalized music recommendation system using a convolutional neural networks approach, Applied Sciences 8 (7) (2018) pp. 1103–1105.

[34] D. Bogdanov, P. Herrera, How much metadata do we need in music recommendation? a subjective evaluation using preference sets., in: ISMIR, (2011), pp. 97–102.

[35] D. Wang, S. Deng, X. Zhang, G. Xu, Learning to embed music and metadata for context-aware music recommendation, World Wide Web 21 (5) (2018) pp. 1399–1423.

[36] P. Knees, M. Schedl, A survey of music similarity and recommendation from music context data, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 10 (1) (2013) pp. 1–21.

[37] T. Baldwin, M. Lui, Language identification: The long and the short of the matter, Association for Computational Linguistics, (2010), pp. 229–237.

[38] J. Schwenninger, R. Brueckner, D. Willett, M. E. Hennecke, Language identification in vocal music., in: ISMIR, Citeseer, (2006), pp. 377–379.

[39] Y. Lei, L. Ferrer, A. Lawson, M. McLaren, N. Scheffer, Application of convolutional neural networks to language identification in noisy conditions, Proc. Odyssey-14, Joensuu, Finland 41 (1) (2014) pp. 1–8.

[40] S. Ganapathy, K. Han, S. Thomas, M. Omar, M. V. Segbroeck, S. S. Narayanan, Robust language identification using convolutional neural network features, in: Fifteenth annual conference of the international speech communication association, (2014).

[41] Jupyter notebook, `https://jupyter.org/`.

[42] Bigquery, `https://cloud.google.com/bigquery/`.

[43] Pandas, https://pandas.pydata.org/.

[44] Numpy, http://www.numpy.org/.

[45] Q. Wei, R. L. Dunbrack, Jr, The role of balanced training and testing data sets for binary classifiers in bioinformatics, PloS one 8 (7).

[46] I. Aryes, Super Crunchers: Why Thinking–By–Numbers Is The New Way To Be Smart., Bantam Books, (2007).

[47] Detectlanguage, https://detectlanguage.com/.

9. Appendix

Song name:

Accuracy: 71.9%

F-score (micro): 0.72

Feature coverage: 100%

Language	*Precision*	*Recall*	*F-score*	*Support*
de	0.88	0.78	0.83	3986
es	0.72	0.74	0.73	3979
fi	0.82	0.75	0.78	4050
fr	0.78	0.66	0.71	4059
it	0.57	0.68	0.62	3948
ja	0.47	0.77	0.58	3946
nl	0.86	0.70	0.77	3960
pt	0.83	0.70	0.76	3958
sv	0.83	0.70	0.76	4114

Table 25: Results for song name in a random forest model with 150 trees for the whole data set.

Album name:

Accuracy: 68.2%

F-score (micro): 0.68

Feature coverage: 100%

Language	*Precision*	*Recall*	*F-score*	*Support*
de	0.81	0.71	0.76	4023
es	0.55	0.75	0.63	3991
fi	0.85	0.81	0.83	3966
fr	0.63	0.58	0.61	3934
it	0.65	0.55	0.60	4016
ja	0.44	0.72	0.55	3969
nl	0.82	0.69	0.75	3976
pt	0.81	0.63	0.71	4065
sv	0.86	0.70	0.77	4060

Table 26: Results for album name in a random forest model with 150 trees for the whole data set.

Track vectors:

Accuracy: 97.4%

F-score (micro): 0.97

Feature coverage: 48%

Language	*Precision*	*Recall*	*F-score*	*Support*
de	0.96	0.94	0.95	1416
es	0.96	0.97	0.96	2007
fi	0.99	1.00	1.00	2310
fr	0.97	0.94	0.96	2349
it	0.94	0.97	0.96	2133
ja	0.99	0.96	0.98	1195
nl	0.99	0.99	0.99	2486
pt	0.99	0.98	0.98	2181
sv	0.98	0.99	0.98	1291

Table 27: Results for track vectors in a random forest model with 150 trees for the whole data set.

Playlist vectors:

Accuracy: 95.1%

F-score (micro): 0.95

Feature coverage: 71%

Language	*Precision*	*Recall*	*F-score*	*Support*
de	0.94	0.92	0.93	2905
es	0.96	0.96	0.96	2915
fi	0.99	0.99	0.99	3293
fr	0.91	0.90	0.90	2948
it	0.88	0.92	0.90	2650
ja	0.96	0.95	0.95	2003
nl	0.98	0.98	0.98	3509
pt	0.97	0.96	0.96	3060
sv	0.97	0.98	0.97	2167

Table 28: Results for playlist vectors in a random forest model with 150 trees for the whole data set.

User vectors:

Accuracy: 94.2%

F-score (micro): 0.94

Feature coverage: 79%

Language	*Precision*	*Recall*	*F-score*	*Support*
de	0.91	0.91	0.91	3470
es	0.95	0.95	0.95	3222
fi	0.99	0.99	0.99	3405
fr	0.89	0.89	0.89	3401
it	0.86	0.88	0.87	3125
ja	0.98	0.96	0.97	2138
nl	0.97	0.98	0.97	3664
pt	0.98	0.96	0.97	3605
sv	0.97	0.97	0.97	2329

Table 29: Results for user vectors in a random forest model with 150 trees for the whole data set.

Regional popularity:

Accuracy: 87.7%

F-score (micro): 0.88

Feature coverage: 86%

Language	*Precision*	*Recall*	*F-score*	*Support*
de	0.87	0.82	0.84	3584
es	0.85	0.85	0.85	3581
fi	0.98	0.98	0.98	3432
fr	0.85	0.79	0.82	3552
it	0.77	0.75	0.76	3366
ja	0.76	0.94	0.84	3402
nl	0.95	0.95	0.95	3709
pt	0.94	0.90	0.92	3760
sv	0.96	0.93	0.95	2500

Table 30: Results for regional popularity in a random forest model with 150 trees for the whole data set.

Genre:

Accuracy: 95.2%

F-score (micro): 0.95

Feature coverage: 62%

Language	*Precision*	*Recall*	*F-score*	*Support*
de	0.91	0.92	0.92	2706
es	0.94	0.96	0.95	2178
fi	0.99	0.99	0.99	2841
fr	0.92	0.92	0.92	2764
it	0.93	0.93	0.93	3117
ja	0.96	0.92	0.94	1686
nl	0.97	0.98	0.97	2673
pt	0.98	0.98	0.98	2486
sv	0.98	0.98	0.98	1878

Table 31: Results for genre in a random forest model with 150 trees for the whole data set.