MLVU final report information sheet

Please include this page in your report either at the start or at the end, before the appendix. Do not change the formatting.

Group number 112

Authors

name	student number
Nasreddine Sahla	2776905
Djio Bartels	2783648
Tunahan Ozturk	2781737
Lisa Keesom	2766152
Enea Camishi	2740935

Software used
We used Python and VSCode to build the models, also including libraries like NumPy, scikit-learn, and Pandas.

Use of AI tools
We created a severity-ranked list of often-used profanity and slurs in songs using ChatGPT. The ranking was low, medium, and high.

Link to code (optional)
https://github.com/EnusaC/ML-Group-112

Group disagreements
*If there are any disagreements in the group, you may **not** remove a student from the author list without their consent. You should bring disagreements to our attention early. As a last resort, you can describe any grievances here.*

Abstract
This study aims to measure the level of influence a song's lyrics have on predicting whether is explicit. By using a dataset of songs with their lyrical content and their corresponding musical features and embedding them into a predictive model for the algorithms to use, the aim is to create a detailed but comprehensive approach to predicting song explicitness. The performance of four different algorithms has been measured on two different versions of the dataset: the dataset with 13 musical features (such as danceability, valence, etc.) versus the dataset that contains the corresponding music features. The performances of the k-Nearest Neighbors, Naïve Bayes, Logistic Regression, and Decision Tree algorithms have been compared with both versions of the dataset. The k-Nearest Neighbors algorithms' performance proved to be the best. The algorithms that were used in this study, if they are optimized properly, can have a lot of practical applications as far as being implemented by labels.

Keywords: Machine Learning, Spotify, Explicitness, Supervised Learning.

1. Introduction

1.1 Background
Over the decades, music has become a lot more accessible for all people around the world. Before the digital age, music, explicit or not, was sold in stores where a hard copy was given to people by people where explicitness could be enforced to people whose explicit songs are not made for, for example, children. In today's digital age, this enforcement has become almost non-existent where children can easily make new accounts on Spotify and listen to any songs they like. This issue is a challenge for both the music industry and society where concerns about the appropriateness of song content have become increasingly pertinent. To label a song explicit is not always straightforward. As most people would think, an explicit song is explicit if it contains swear words. However, this is not always the case. If the artist talks about something that is otherwise considered a swear word but does not mean it to be a swear word but to be used as an actual normal word in a sentence, it could be argued that the song is not explicit. Other features also contribute to a song's explicitness and some features contribute more than others where a song that has an uplifting beat, rhythm, instrumental, etc. is less likely to contain such explicit words that are meant as explicit words i.e., musical features, other than its lyrics, also contribute to the overall impression of a song and its impact it has on its listeners. An algorithm that can predict a song's explicitness by looking at its musical features can help labels and lawmakers since an algorithm of the sort is more specific and clearer than humans where humans would have different opinions on which words in which context could mean that a song is explicit or not. This algorithm can use more detailed data such as a song's musical features to determine whether a song is explicit or not. This helps labels and lawmakers with efficiency and time but also a more precise model that labels songs explicit or not which can help parents by giving them a precise list of non-explicit songs that can be listened to by children or not. The objective of this study is to develop an algorithm that looks at musical features and tries to predict where a song is likely to be explicit or not based on those musical features. By using a dataset of songs with their lyrical content and their corresponding musical features and embedding them into a predictive model for the algorithm to use, the aim is to create a detailed but comprehensive approach to predicting a song's explicitness.

1.2 Dataset Description
The original dataset stems from Kaggle and is posted by the user Joakim Arvidsson [1]. The dataset is a CSV file with almost 30000 songs retrieved from the Spotify API. Alongside the song name, there are also other characteristics included, such as track artist, track popularity, and track album id. There also are columns referring to 13 musical features of each song:

• Danceability
• Energy
• Key
• Loudness
• Mode
• Speechiness
• Acousticness
• Instrumentalness
• Liveness
• Valence
• Tempo
• Duration ms

These features are expected to be quite informative for the use case of predicting explicitness, for example, if a song is long (high duration_ms) it might indicate it is a classical play, which often is not explicit.

2. Data Inspection and Preparation

Before the dataset is usable for the use case of predicting whether a song is explicit, several adjustments need to be made. The current dataset does not contain whether a song is explicit, which is necessary for labeling.

2.1 Data Inspection

Initial dataset: The initial dataset is described in section 1.2. However, this dataset has shortcomings when it comes to essential features like explicitness and no data regarding the lyrics of the songs. Lack of information about the lyrics is detrimental to the use case, so something must be done about it. Without data on the lyrics, the research question can never be answered. Added features: To adjust the initial dataset, the following adjustments have been made:
• Added explicitness flag
• Added lyrical explicitness score.

The Spotify API has been accessed using the *spotipy python package* [2] and iterated over all instances in the dataset to retrieve whether a song was explicit. If it was, it got flagged as True, and if it was not, it got flagged as False.
To add a lyrical explicitness score for each song, the lyrics of each song had to be accessed. This has been done by accessing the Genius API (Genius is the world's biggest collection of song lyrics and musical knowledge) using the *lyricsgenius python package* [3]. The whole dataset has been iterated over to add the lyrics for each song. To do this without getting any (formatting) errors, the dataset had to be reduced by using some basic filters, such as removing empty lyrics, removing audiobooks, etc. Now that all songs got their lyrics, a script has been written based on a huge variety of curse words, like "fuck" and "bitch", and various slurs provided by ChatGPT. These words would weigh more than words like "shoot", which has a lower weight since it can be used in other non-explicit contexts as well. These weighted curse words and slurs gave a lyrical explicitness score that the machine learning algorithms will be learning from. Due to some errors, we encountered in lyrics and Spotify rate limits, it was not possible to optimize the size of the dataset, so, in the end, the dataset was left with 16000 songs with the explicitness flag and lyrical explicitness score.
By adding the explicitness flag and lyrical explicitness score to the initial dataset, an improved dataset has been achieved. This new dataset is better suited for the use case.

2.2 Dataset Preparation

There were only 4239 explicit songs out of the 16000 songs in the new dataset, which created a significant imbalance. According to lecture 3, this can lead to problems such as bias towards the majority class, poor generalization, and misleading evaluation metrics. In section 3, the optimal course of action under this imbalance is discussed. The dataset is split into a training set of 13000 and a test set of 3000.

3. Methodology

The performance of four different algorithms has been measured on two different versions of the dataset: the dataset with the 13 musical features (such as danceability, valence, etc.) versus the dataset with the lyrical explicitness score. The following four algorithms have been used:

- K-Nearest Neighbors (kNN)
- Logistic Regression
- Decision Trees (DTs)
- Naïve Bayes

Before starting the measurements, k-Nearest Neighbors was expected to have the best performance score since this algorithm seemed best suited for this dataset because kNN makes predictions based on the similarity of instances in the feature space. In this dataset, where songs are represented by a set of musical features, kNN can effectively use the similarity between songs to make predictions. Due to the dataset being imbalanced, the accuracy of the algorithms alone is insufficient to assess their performance. According to lecture 3, the best thing to do under class imbalance is to look at the performance in more detail, and this can be achieved by using the following metrics:

- Accuracy
- Precision
- Recall
- False Positive Rate

The measurements have been coded using Python.

3.1 k-Nearest Neighbors (kNN)

The first algorithm was the k-nearest neighbors (KNN) algorithm, which is a lazy supervised algorithm that uses proximity to make classifications or predictions about the grouping of individual data points in the feature space. It is one of the popular classification and regression classifiers used in machine learning today [4]. KNN being a lazy algorithm means that it does not do any learning. It just remembers the data (Lecture 1). This algorithm has been chosen because it is easy to understand, and it is an algorithm that has appeared throughout the AI bachelor a considerable number of times. Implementing this algorithm in Python did not prove to be difficult, because the required information to implement this was gathered through the scikit-learn [5] documentation and IBM [4]. Selection of the hyperparameter (k) was done in two ways:

1. Grid search
2. Simulated annealing

First, both versions of the dataset have had their hyperparameter selected using grid search with a range of 350, and then both versions of the dataset have had their hyperparameter selected using simulated annealing with a range of 1000. Both methods have their advantages and disadvantages, such as grid search being an exhaustive search and having a straightforward implementation [6] but falling short when it comes to its computational intensity. Simulated annealing explores the search space more intelligently compared to brute-force methods like grid search, and it is adaptable to large search spaces, but it has parameter tuning complexity because it involves tuning parameters such as the initial temperature, cooling schedule, and acceptance probability [7]. Both selection methods being so different from each other is the reason for choosing both.

3.2 Logistic Regression

The second algorithm was logistic regression, which is a discriminative classifier that learns to map the features directly to class probabilities, without using Bayes' rule to reverse the conditional probability. It is basically a small extension of the linear classifier, and it can also be thought of as a linear classifier with a specific loss function (definition from lecture 4). This algorithm has been chosen because a few group members had a deep understanding of this algorithm, which made it a great choice. Implementing this in Python went without any issues since the scikit-learn documentation was available [8]. The only thing that caused a bit of trouble was deciding on what hyperparameters to select. Ultimately, the decision of not changing any of the hyperparameters has been taken. The default scikit-learn hyperparameters have been utilized, as logistic regression can perform reasonably well with these default settings.

3.3 Decision Trees

Following the Logistic regression algorithm, the focus shifted to decision trees. Decision trees (DTs) are described as a 'non-parametric supervised learning method', emphasizing their ability to predict target values without strict assumptions about the underlying data distribution. The decision to implement this method was motivated by the ease of interpretation as the trees can be visualized. This makes understanding the decision-making process more linear. Furthermore, the ability for DTs to handle both numerical and categorical data. This would be exceptionally beneficial when dealing with heterogeneous datasets with diverse types of information such as beats per minute (BPM) and music genres. Finally, decision trees require minimal data preprocessing, simplifying the implementation process [9]. However, despite these advantages, DTs are not frequently used due to their flawed tendency of overfitting – which can't be improved without decreasing the method's performance quality (Lecture1).

The reason to implement the method, even with its risks, is found in the curiosity to see how such a method would perform against other algorithms and hoping to find compelling findings. And, as previously described, the need for little data preparation made for easier execution. Like kNN and Logistic regression, the Decision trees were implemented with the help of the scikit-learn documentation. The implementation itself went smoothly. Still, understanding how entropy worked with decision trees was more challenging than initially thought. In the context of decision trees, entropy serves as a criterion for determining where the tree nodes should split, which would eventually help reach better predictions. After getting a better grasp of the formula applied for entropy, the grasp around decision trees improved overall.

3.4 Naïve Bayes

Naïve Bayes classifiers, unlike Decision Trees, belong to the supervised family of probabilistic classifiers based on Bayes' theorem with an assumption of independence among predictors. This method holds its ground in simplicity and efficiency, making it a popular choice for various classification tasks [10]. The motivation behind implementing Naïve Bayes stemmed from its swift execution and decent performance across diverse datasets, even with its simplifying assumption. This simplicity offers a clear advantage, particularly when dealing with large datasets. One of the key reasons to use Naïve Bayes is that it's very robust against overfitting.

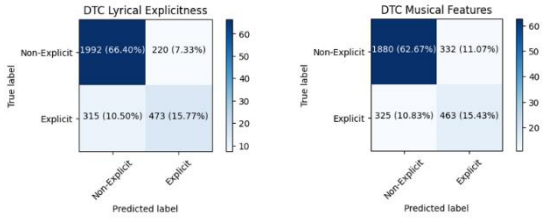
a common challenge faced by algorithms like Decision Trees. Due to its simplicity and efficient use of data, Naïve Bayes tends to generalize well even with limited training instances, making it suitable for scenarios with differing sizes of datasets, from small to large.

Still, Naïve Bayes also has its limitations. For example, each feature within Naïve Bayes contributes to the probability of certain outcomes independently from one another. This could result in suboptimal performance. The reliance of Naïve Bayes on prior probabilities might create biases as well. Again, scikit-learn documentation was used and studied to implement the Naïve Bayes algorithm. Its comprehensive information made understanding the methods applied to the algorithm feel effortless and little to no drawback was met. It is known that Naïve Bayes can tend to overfit if the dataset it's working with is particularly large. However, issues much like these were fortunately avoided.

4. Results

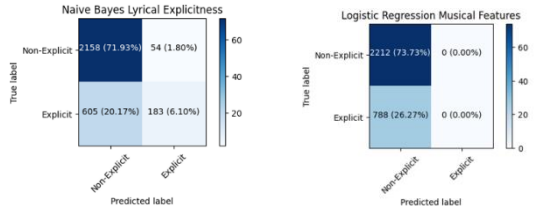
Results were obtained from four different machine learning models for this task and there are a total of 10 results. For one model (k-Nearest Neighbors) four separate tests were conducted. These four tests were a mix of two different inputs (1. Using isolated musical features such as loudness, valence, danceability, etc. 2. Using a lyrical explicitness score.) and two different ways to define K. (1. Grid search. 2. Simulated annealing). The remaining models, namely the Decision Tree Classifier, Naïve Bayes Classifier, and Logistic Regression were each applied twice – once utilizing the musical features and once with the lyrical explicitness score. From each test, the results came out in the form of a confusion matrix, including values for its accuracy, precision, recall and false-positive rate.

4.1 Model Performance



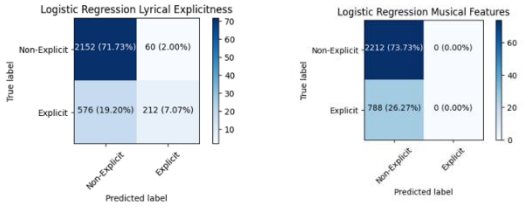
Decision Tree Classifier:

For this classifier, the lyrical explicitness (LE model was more successful than the musical features (MF) one, due to the higher accuracy (LE 0.82 vs 0.78 MF) precision (LE 0.68 vs 0.58 MF), and recall (LE 0.6 vs 0.59 MF). Both models had relatively similar false positive rates.



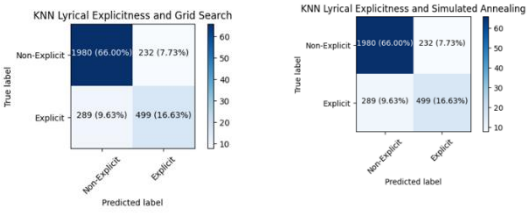
Naïve Bayes:

For the Naïve Bayes Classifier, the model performed similarly across both tests in terms of accuracy (LE 0.78 vs 0.74 MF); however, the musical features model believes all songs were non-explicit. This means that while the accuracy was pretty good, the model completely failed to predict explicit songs based on musical features. For precision, recall, and FP rate, the LE model received 0.77, 0.23, and 0.02 respectively and the MF model received 0.00 for all three.



Logistic Regression:

Logistic Regression models had similar results as the results of Naïve Bayes, with decent accuracy (LF 0.78 vs 0.74 MF) The model had higher precision (0.78), and recall (0.27) when predicting based on lyrical explicitness and failed when based on musical features (0.00). The low recall rate also points towards this model being less efficient.



k-Nearest Neighbors:

kNN provided the most balanced results. The method for defining K (either finding it through grid search or by simulated annealing) did not change anything for the model, because both ways yielded the same results. The Lyrical Explicitness model had much more impressive results than the musical features model. Their accuracy (LF 0.83 vs 0.74 MF), precision (LF 0.68 vs 0.64 MF), and especially their recall (LF 0.63 vs 0.02 MF), were much higher.

4.2 Feature Importance

When comparing the performances, the lyrical explicitness score method tended to outperform musical features. Almost every single LE model had a higher accuracy score than its MF counterpart. Overall, the models' precision ranged between 0.54 and 0.78, which indicates a moderate ability to correctly classify explicit songs.

The recall rates, however, were questionable, as many models had low rates and the highest one was 0.6. This could mean that there was difficulty in highlighting all explicit songs in the dataset. Lastly, it should be mentioned that the dataset had quite a notable imbalance between classes, as there were many more non-explicit songs. This is reflected in the models' overall low false positive rate, with an average of 0.05. It could be implied that there was a predisposition towards predicting songs as non-explicit. Overall, using lyrical explicitness proved to be a better way to predict explicitness in a song.

4.3 Model Comparison

Overall, the models were not the best for the task, as many of them had very low recall and precision rates. Some other models also were unable to predict songs as explicit, rendering them useless for this task. The best possible model for this task was one that was already expected to work well with this study, and that is the k-Nearest Neighbors with the Lyrical Explicitness score. It provided a good balance of correctly predicting scores while minimizing false markings. With an accuracy of 0.83, the model was the best-performing out of the 10 tests. However, it should also be noted that there is room for improvement, as the precision and recall were underwhelming (0.68 and 0.63).

5. Discussion

5.1 Analysis

After finding which model is the best, it is important to understand why it was the best. The model that performed the best out of the four algorithms applied was the kNN model. This is probably because of the difference in specific aspects the kNN algorithm has that the other algorithms do not have, especially with very vague features in the dataset that do not give a noticeable difference between whether a song is explicit or non-explicit. These vague features made it harder for the other algorithms while it did not affect the kNN algorithm as much as it has the best accuracy out of the 4 algorithms. kNN does this in multiple ways such as being robust to irrelevant features. Many features in the dataset are irrelevant and do not give any correlation between a song being explicit or non-explicit. kNN is robust to these features whereas other algorithms such as Decision Trees and Naïve Bayes did have some problems with those features. The ROC curve also showed that kNN is the best model and how the lyrics are still very important to labeling songs explicit or not than the musical features of the song which makes a lot of sense. In the end, it is the words that can make a song explicit and not the musical features. The lyrical explicitness score that was implemented also helped Logistic Regressions. The score was calculated by the amount of different curse words in a song and then weighing certain curse words more than others. The score helped Logistic Regression because Logistic Regression works in a binary outcome, in this case, explicit or non-explicit. It does this based on the weighted sum of input features and by using the lyrical explicitness score gives the algorithms one of the most valuable features to predict if a song is explicit or not, namely explicitness. So, by using this lyrical explicitness score, it could predict songs better on whether they would be explicit or not.

5.2 Limitation

When working with various algorithms, their performances differ when dealing with various limitations. These limits come in various categories, they may stem from the algorithm itself or the dataset they work with for example.

5.2.1 Class Imbalance

This is a limitation experienced throughout researching which algorithm can predict song explicitness most accurately. Class imbalance comes down to having an imbalance in variables. Examples have been mentioned before during lectures with women who do or don't have breast cancer. There are (fortunately) far more healthy women than women who have breast cancer. However, for a database to be reliably trained, such an imbalance is unbeneficial. Class imbalance is something that was experienced with the song dataset as well.

There is about a 75% to 25% ratio between songs that are not explicit versus songs that are explicit on Spotify, respectively. This is something that was taken into consideration throughout the trajectory of training to testing.

5.2.2 Overfitting

Overfitting takes place when a model learns patterns from a certain dataset too tightly knit to its training set. While the algorithm applied to a dataset will be able to accurately state the matching information about the instances within the training set, it is likely to misclassify and get confused when presented with new data [11]. This is a frequent problem with machine learning models. However, it is safe to say that overfitting has been mitigated with the four algorithms tested. A possible reason for this is that by including songs from the database that had their lyrics missing, it was 'forced' to generalize the data, which would reduce the chances of overfitting.

5.2.3 Quality and Accuracy

As mentioned, a part of the data within the dataset was missing the lyrics to their songs. Reasons for this could be that Genius, "the biggest collection of song lyrics and musical knowledge", still misinterprets many songs' lyrics. So, the decision was made to exclude these lyrics from the dataset to preserve the quality of the data used. Due to the sheer size of the dataset, it would have been far too time-consuming and strenuous to manually add accurate lyrics to the dataset, which is another reason for missing lyrics within the data. While the reduced number of lyrics may have reduced chances of overfitting, it could also have the polarizing effect that it could lead to inaccuracies with the predictions various models may produce.

5.3 Future Works

Moving forward, future research could focus on enhancing the accuracy of explicit content detection on songs using the algorithms employed in this study (kNN, Decision Trees (DTs), Naïve Bayes, and Logistic Regression). One critical aspect for improvement is addressing the limitations of the dataset. The dataset used in this study was missing a lot of lyrics for many songs, potentially impairing the accuracy of the classification models. Future research could involve collecting or augmenting datasets with more comprehensive lyric information to improve overall model performance. Additionally, the dataset had a substantial class imbalance, with only 25% of songs labeled as explicit, as opposed to the 75% not explicit. Balancing the dataset or using techniques such as oversampling could prevent class imbalances in future versions of this study. By addressing these limitations, future studies can refine the performance of the algorithms and contribute to more
accurate and reliable explicit content detection on music streaming platforms. Another point of interest could be to use other models instead of the ones currently used. Many comparisons are still untapped and are interesting to explore.

6. Conclusion

The main key finding in this study was that explicit and non-explicit songs do not have specific data in their musical features that could differentiate them from each other. Except for the lyrical explicitness score that was made specifically for this study, songs have roughly the same data for their musical features and thus it's hard to indicate if a song is explicit or non-explicit. The features that were in the dataset gave very vague and not very usable data to make a model out of. Most of the algorithms applied did have problems because of the vagueness, whereas the kNN algorithm had the least number of problems with this data and predicted if a song was explicit or not the best. This could be an issue with the data and the musical features, but it could also be that there is no data in musical features that differentiates explicit songs from non-explicit songs. If it is an issue with the data, the algorithms could be improved in the future if the dataset is also improved upon. The model that was produced in this study, if it is optimized properly, can have a lot of practical applications as far as being implemented by labels to label songs as explicit or not based on other features than only on the lyrics. This form of labeling using the algorithm will be more time-efficient and practical as it would label songs faster and with fewer misclassifications. It can also be used in recommendation systems

where it will recommend explicit or non-explicit songs based on what it asked and not exclude songs that are only explicit based on their lyrics if the artist did not intend for them to be explicit but used the words for artistic value. Lastly, it could be used as content filtering by radio stations, streaming services, etc. where it would automatically filter explicit content/songs so that the users are only exposed to content that the media wants the users to be exposed to.

References

[1] J. ARVIDSSON, “30000 Spotify songs,” 2023. https://www.kaggle.com/datasets/joebeachcapital/30000-Spotify-songs.

[2] “Welcome to spotipy! - spotipy 2.0 documentation,” spotipy.readthedocs.io. https://spotipy.readthedocs.io/en/2.22.1/.

[3] “Lyricsgenius: a python client for the genius.com,” lyricsgenius.readthedocs.io., https://lyricsgenius.readthedocs.io/en/master/.

[4] IBM, “What is the k-nearest neighbors algorithm?,” 2023. www.ibm.com, https://www.ibm.com/topics/knn.

[5] “1.6. nearest neighbors — scikit-learn 0.21.3 documentation,” 2019. Scikit-learn.org, https://scikit-learn.org/stable/modules/neighbors.html.

[6] “scikit-learn, “sklearn.modelselection.gridsearchcv|scikit−learn0.22documentation,″ 2019.Scikit−learn.org, https : //scikit−learn.org/stable/modules/generated/sklearn.modelselection.GridSearchCV.html.

[7] G. C. D. Kirkpatrick, S. and M. P. Vecchi, “Optimization by simulated annealing. science, 220:671-680.,” 1983.

[8] “1.1. linear models — scikit-learn 0.22.2 documentation,” Scikit-learn.org, https://scikit-learn.org/stable/modules/linearmodel.htmllogistic − regression.

[9] “1.10. decision trees,” 2019. Scikit-learn. https://scikit-learn.org/stable/modules/tree.html.

[10] J. Galef, “A visual guide to bayesian thinking,” 2015, July 16. YouTube, https://www.youtube.com/watch?v=BrK7XX lGB8.

[11] X. Ying, “An overview of overfitting and its solutions,” In Journal of physics: Conference series, vol. 1168, p. 022022, 22019, February.

[12] L. Bergelid, “Classification of explicit music content using lyrics and music metadata,” 2018.