Rockify — Rock Music Subgenre Classifier

A Machine-Learning Practice using Random Forest Classification Algorithm and Comparative Analysis with Other Models

- Contributors: Yvonne Yang, Jiahui Gao, Emily Chou, Chia-Wei Chang
- Course: Managing Data & Signal Processing
- Instructor: Sep Makhsous
- Department: Master of Science in Technology Innovation, the University of Washington

Table of Contents

1. Introduction
2. Methods
3. Results
4. Discussion
    - Result Analysis — Why is Random Forest the Best for Rockify
    - In-Depth Result Analysis — Random Forest vs. Decision Tree
    - In-Depth Result Analysis — Why K-Nearest Neighbors Do Not Work Well?
    - In-Depth Result Analysis — Why Support Vector Machines Do Not Work Well?
    - In-Depth Result Analysis — Why Logistic Regression Do Not Work Well?
    - Limitation — Sub-Genre Dataset
    - Future Direction — Use Fewer Features & Build a Model with Better Accuracy
    - Rockify Source Code (Github)
5. Conclusion
6. References
7. Appendix

Introduction

This project aims to develop a machine-learning model for music genre classification to identify various rock music sub-genres. Our model, Rockify, is trained on a dataset of rock music from Spotify to classify songs into specific sub-genres, including but not limited to classic rock, alternative, and heavy metal. In this report, we will guide you

comprehensively through the methods used, the model, the results, and an in-depth discussion of the implications of this breakthrough in music technology.

Methods

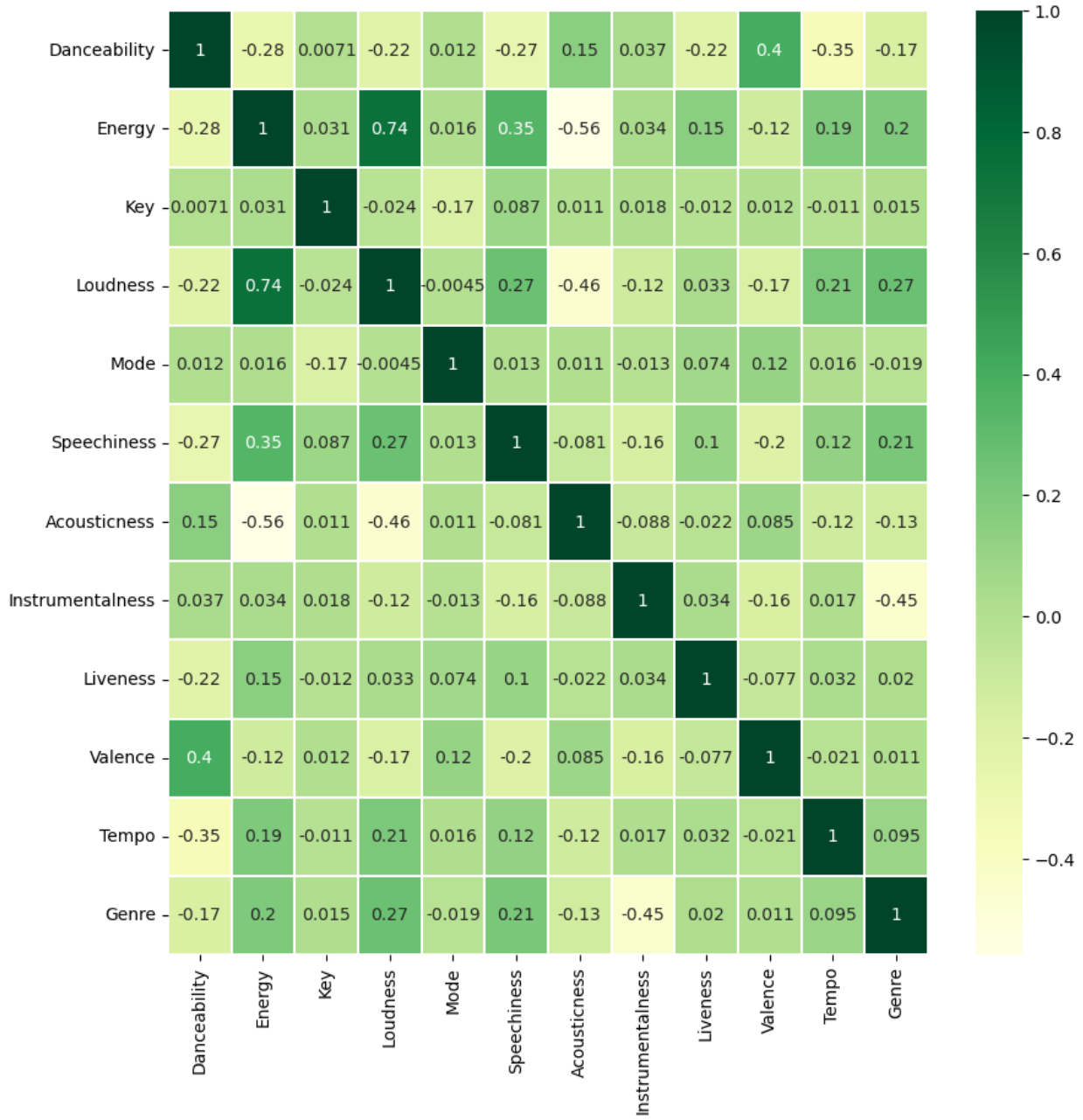
In order to develop Rockify, we employed various machine-learning algorithms to identify the best-performing model. Firstly, we began by obtaining a dataset of rock songs from Spotify API, consisting of thousands of songs in various sub-genres of rock music. Soon after, we experimented with several machine learning algorithms, including Decision Tree Model, KNN Model, SVM Model, Logistic Regression Model, and Random Forest Model. After evaluating the performance metrics of each algorithm, we found that the Random Forest Model had the best performance with an accuracy of approximately 0.55, so we decided to use the Random Forest Model.

To implement the algorithms and conduct our analysis, we utilized MATLAB and Python programming languages. To go in-depth, we employed the scikit-learn library for implementing the Decision Tree and Artificial Neural Network algorithms, and the RandomForestClassifier algorithm from scikit-learn for implementing the Random Forest model. Through our comprehensive experimentation and analysis of various machine learning algorithms, we have successfully developed Rockify, a powerful tool for classifying songs into sub-genres of rock music.

Results

After extracting the data, we performed a correlation analysis on 11 audio features (Danceability, Energy, Key, Loudness, Mode, Speechiness, Acousticness,

Instrumentalness, Liveness, Valence, and Tempo) obtained through the Spotify API, as well as a manual classification tag (sub-genre generated from playlists).



Correlation analysis is performed on the independent variables because if there is a strong correlation, they may provide replication or almost identical information, leading

to overfitting the model. In this case, dimensionality reduction of the independent variables can be considered to remove redundant information and improve the model.

We found that the random forest model had the highest accuracy of all the models. Each decision tree focuses on only a subset of features chosen randomly. The size of this feature subset is usually much smaller than the size of the original feature set, so it can be seen as an implicit dimensionality reduction method.

The following results were fitted using Decision Tree Model, Random Forest Model, KNN Model, SVM Model, and Logistic Regression Model, respectively:

Decision Tree Classifier Model

Accuracy: 0.45989304812834225

Confusion Matrix:

```
[[22  3  4  2  1]
 [ 8 25 13  1  2]
 [ 2  5  8 28  0]
 [ 3  3 20 13  0]
 [ 3  3  0  0 18]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.58	0.69	0.63	32
1	0.64	0.51	0.57	49
2	0.18	0.19	0.18	43
3	0.30	0.33	0.31	39
4	0.86	0.75	0.80	24
accuracy			0.46	187
macro avg	0.51	0.49	0.50	187
weighted avg	0.48	0.46	0.47	187

Random Forest Classifier Model

Accuracy: 0.5935828877005348

Confusion Matrix:

```
[[23  3  3  3  0]
 [ 2 41  3  1  2]
 [ 1  5 10 27  0]
 [ 1  5 17 16  0]
 [ 0  2  0  1 21]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.85	0.72	0.78	32
1	0.73	0.84	0.78	49
2	0.30	0.23	0.26	43
3	0.33	0.41	0.37	39

	4 0.91 0.88 0.89 24 accuracy 0.59 187 macro avg 0.63 0.61 0.62 187 weighted avg 0.59 0.59 0.59 187

K-Nearest Neighbors Classifier Model

Accuracy: 0.32085561497326204	
Confusion Matrix: [[11 10 8 2 1] [12 28 3 5 1] [4 14 7 14 4] [10 10 7 10 2] [2 6 10 2 4]]	Classification Report: precision recall f1-score support 0 0.28 0.34 0.31 32 1 0.41 0.57 0.48 49 2 0.20 0.16 0.18 43 3 0.30 0.26 0.28 39 4 0.33 0.17 0.22 24 accuracy 0.32 187 macro avg 0.31 0.30 0.29 187 weighted avg 0.31 0.32 0.31 187

Support Vector Machine (SVM) Classifier Model

Accuracy: 0.28342245989304815	
Confusion Matrix: [[4 8 12 8 0] [8 29 4 7 1] [2 13 4 21 3] [3 11 11 13 1] [0 6 9 6 3]]	Classification Report: precision recall f1-score support 0 0.24 0.12 0.16 32 1 0.43 0.59 0.50 49 2 0.10 0.09 0.10 43 3 0.24 0.33 0.28 39 4 0.38 0.12 0.19 24 accuracy 0.28 187 macro avg 0.28 0.25 0.24 187 weighted avg 0.27 0.28 0.26 187

Logistic Regression Classifier Model

Accuracy: 0.43315508021390375

Confusion Matrix:

```
[[19  6  1  5  1]
 [ 2 32  3 10  2]
 [ 1 19  6 11  6]
 [ 2 11  4 14  8]
 [ 0  8  1  5 10]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.79	0.59	0.68	32
1	0.42	0.65	0.51	49
2	0.40	0.14	0.21	43
3	0.31	0.36	0.33	39
4	0.37	0.42	0.39	24
accuracy			0.43	187
macro avg	0.46	0.43	0.42	187
weighted avg	0.45	0.43	0.42	187

From the above results, it can be seen that the random forest classification model is significantly better than other methods. Therefore, we further optimized this model. We used the GridSearchCV function to perform a grid search. In the end, we increased the accuracy by a few percent.

Random Forest Classifier Model (after optimization)

One of the Best Hyperparameters Found:
bootstrap = False; class_weight = 'balanced'; criterion = 'entropy'; max_depth = 5; max_features = 'sqrt'; min_samples_leaf = 1; min_samples_split = 5; n_estimators = 100

Accuracy: 0.6310160427807486

Confusion Matrix:

```
[[24  4  1  2  1]
 [ 2 38  5  1  3]
 [ 3 11 18  9  2]
 [ 3  7 13 15  1]
 [ 0  1  0  0 23]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.75	0.75	0.75	32
1	0.62	0.78	0.69	49
2	0.49	0.42	0.45	43
3	0.56	0.38	0.45	39
4	0.77	0.96	0.85	24
accuracy			0.63	187
macro avg	0.64	0.66	0.64	187
weighted avg	0.62	0.63	0.62	187

Finally, we wrote a convenient query program for users. In our program, entering the song URL from Spotify can get genre analysis results. For example:

```
Please enter the song URL:
https://open.spotify.com/track/1RxLSgcsZimzKnpCLzdVsi?si=0f467b26c4f84abd
[Rockify] The Song's Track ID is:  1RxLSgcsZimzKnpCLzdVsi
[Rockify] This song likely belongs to: Punk Rock
```

Discussion

Result Analysis — Why is Random Forest the Best for Rockify

Random forest classifier is an ensemble learning technique combining multiple decision trees to improve the model performance and reduce overfitting. In this model, numerous decision trees are trained on different subsets of the training data. The final prediction is made by taking an average or majority vote over the individual decision trees.

Compared to other models in our project, the Random Forest Classifier model produced the highest accuracy score of around 0.6. This may be because random forest classifiers are less prone to overfitting than decision tree and K nearest neighbor models. Also, it can handle nonlinear relationships between features and target variables better than logistic regression and support vector machine models.

In-Depth Result Analysis — Random Forest vs. Decision Tree

The decision tree classifier model has an accuracy of 0.4599, lower than the random forest classifier. Decision trees are prone to overfitting, and this model may overfit the training data, resulting in a lower accuracy score.

Suppose we have a dataset of rock music tracks with features like tempo, loudness, and energy. We want to classify them into different genres. We can create a classification model for this dataset using a decision tree or random forest model.

If the decision tree model creates a tree with too many splits and leaves, it may overfit the training data. For example, if a decision tree model generates a branch for each music track in the training data, it may be too close to the given data to generalize nicely to new, unseen data. This can result in a lower accuracy score, as shown in our result, with an accuracy of 0.4599.

On the other hand, random forest models create multiple decision trees with different features and subsets of data and combine their predictions. This approach helps reduce overfitting and improves the model's ability to recognize new data. In our rock music sub-genre classification project, the random forest model better captures the relationship between different features and sub-genres, resulting in a higher accuracy score of 0.5936.

In-Depth Result Analysis — Why K-Nearest Neighbors Do Not Work Well?

The K-Nearest Neighbors Classifier Model had a low accuracy score of 0.3209. K-NN is a simple and intuitive algorithm, but it suffers from the curse of dimensionality and performs poorly when the number of features is as large as our project has.

Suppose we have a dataset of rock music songs. We want to classify them into different subgenres, such as classic rock, punk rock, and grunge. We collect several features for each music, such as speed and energy. However, we also decide to include more complex features, such as the length of a guitar solo and the number of time signature changes, like Rockify using 11 features as input to the machine.

When we attempt to apply the K-Nearest Neighbors algorithm on this dataset, we will encounter the curse of dimensionality. The curse of dimensionality states that as the number of features or dimensions in the feature space increases, the amount of data required to generalize accurately increases exponentially. This is because, in high-dimensional feature spaces, the distance between any two points becomes larger and larger, making it difficult to differentiate between them.

As the high number of features in Rockify, the distance between any two points in the feature space increases, and the distance difference becomes negligible. In other words, K-NN works well when the number of features is small (namely, the feature space is not too complex). However, K-NN may not be the best choice for classification tasks when the feature space becomes too large.

In-Depth Result Analysis — Why Support Vector Machines Do Not Work Well?

The accuracy score of the Support Vector Machine (SVM) classifier model was 0.2834, which was lower than the other models. SVMs can perform well on small datasets, but their performance degrades as the size increases. Also, the RBF kernel used in this model may not be suitable for this dataset.

First of all, the principle of the SVM classifier itself makes it difficult for projects like ours that need the support of large datasets. Suppose we have a small dataset of only 50 tracks on which we train an SVM classifier with an RBF kernel. In this case, the SVM classifier might perform pleasingly and give us good accuracy. However, the performance of an SVM classifier may degrade as the dataset size increases because the computational complexity of the SVM algorithm grows rapidly with the number of data points.

In SVM, the hyperplane separating the classes is chosen by maximizing the separation between the classes. The data points closest to the hyperplane are used to define the hyperplane and are called support vectors. As the dataset size increases, the number of support vectors increases, making finding the optimal hyperplane computationally expensive.

Assuming we do not have a computational cost issue, as the dataset size increases, so does the risk of overfitting, leading to poor generalization performance of the SVM model. Also, if the dataset contains features not well separated by the RBF kernel, the SVM classifier may not perform well on such datasets. For example, if different subgenres have features overlapping significantly, the RBF kernel may fail to capture these subtle differences, resulting in a lower accuracy score.

Overall, SVM may not be the best choice for large datasets, and other algorithms, such as Random Forest, may be better suited for our case.

In-Depth Result Analysis — Why Logistic Regression Do Not Work Well?

The logistic regression classifier model has an accuracy of 0.4332, lower than the random forest classifier. Logistic regression assumes a linear relationship between the features and the target variable, which may not hold in our dataset.

In Rockify, the relationship between our input features and subgenres is not strictly linear. For example, a music track with high energy, loudness, and rhythm could be categorized as metal or punk based on factors like key or mode. In such cases, linear models such as logistic regression may not accurately capture the relationship between features and target variables, resulting in lower accuracy scores than non-linear models such as random forests. On the other hand, random forests can

handle non-linear relationships between features and target variables by building decision trees and combining them to create ensemble models.

Limitation — Sub-Genre Dataset

In this project, we aimed to classify rock music into sub-genres using machine learning (ML) in Python. To create a training dataset, we manually searched for existing playlists on Spotify that matched the rock music sub-genres we wanted to predict: (1) E-rock; (2) Classic rock; (3) Funk rock; (4) Metal rock; (5) Punk rock. Although our search yielded playlists that predominantly represented the given rock sub-genres, the accuracy of our model was average and could be improved.

To address this limitation, we plan to adopt the "Delphi technique," which is an established approach to achieving a consensus view across subject experts. By recruiting three rock music experts, such as rock musicians, critics, or fans with extensive experience in listening to a wide range of rock music, we can reduce bias in the labeling of sub-genres in our dataset. The Delphi technique enables participants to reflect on their opinions and refine their views based on the anonymized feedback of others, helping to establish a more accurate classification of the sub-genres.

In addition to the Delphi technique, other ways to improve our model's accuracy could be exploring more advanced ML algorithms or incorporating additional features from the audio signal. Nevertheless, employing the Delphi technique would undoubtedly help us reduce bias and improve the classification accuracy of our rock music sub-genres.

Future Direction — Use Fewer Features & Build a Model with Better Accuracy

In the future, there are several directions we could take to improve the accuracy and efficiency of our rock sub-genre classification project using ML in Python.

Firstly, we could explore the possibility of reducing the number of features used in our model while maintaining high accuracy. Although we currently use 11 features obtained from the Spotify API, it may be possible to achieve similar results with fewer features. By conducting feature selection and evaluating the impact on model performance, we could identify the most important features and build a more streamlined model.

Secondly, we could further leverage the Spotify API by incorporating additional features, such as lyrics or album art, which may provide more valuable information for genre classification.

Lastly, to increase the efficiency of the training model, we could utilize a correlation heat map to identify and remove any features that are not highly related to the target variable. This approach would help us to reduce computational resources and improve the performance of the model. For instance, by using only the eight most highly related features, we could potentially increase the accuracy of our model while reducing the training time.

Conclusion

Rockify has the potential to target these 3 audiences, music producers, streaming platforms, and music enthusiasts, by providing accurate rock music sub-genre classification. First of all, Rockify can help music producers create targeted music that

resonates with their audience, while also discovering new and emerging sub-genres of rock music. By being at the forefront of music trends, producers can position themselves as innovators in the industry and stay ahead of the competition. Secondly, streaming platforms can use Rockify to improve their music recommendations, leading to increased user engagement and satisfaction. And for music enthusiasts, Rockify can be a valuable tool for discovering new music and exploring the diverse sub-genres of rock music.

In conclusion, Rockify showcases the immense potential of machine learning in music genre classification. Our model's accuracy of approximately 0.6 shows the effectiveness of the Random Forest algorithm for this task. However, we believe there is still room for improvement and expansion of our model by exploring more advanced machine learning techniques like deep learning. As we continue to innovate and improve our model, we hope to empower music producers, streaming platforms, and music enthusiasts to make data-driven decisions that enhance the listening experience and create a more diverse and vibrant music industry!

References

[1] Patel, D. (2019). Music genre classification with Python. Towards Data Science. https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8

[2] Spotify for Developers. (n.d.). https://developer.spotify.com/

[3] TOMIGELO. (2019). Spotify audio features [Data set]. Kaggle. https://www.kaggle.com/datasets/tomigelo/spotify-audio-features

[4] Spotipy API Reference. (n.d.). Spotipy. https://spotipy.readthedocs.io/en/2.22.1/#api-reference

[5] Seaborn heatmap. (n.d.). Seaborn. https://seaborn.pydata.org/generated/seaborn.heatmap.html

[6] Kumar, A. (2022). Correlation concepts, matrix & heatmap using Seaborn [Blog post]. Data Analytics. https://vitalflux.com/correlation-heatmap-with-seaborn-pandas/

[7] Andrade, A. (2017). Correlation heatmap [Online forum post]. Stack Overflow. https://stackoverflow.com/questions/39409866/correlation-heatmap

[8] Python [Errno 98] Address already in use [Online forum post]. Stack Overflow. https://stackoverflow.com/questions/38543284/python-errno-98-address-already-in-use

[9] IBM. (n.d.). What is Random Forest? Retrieved March 12, 2023, from https://www.ibm.com/topics/random-forest

[10] IBM. (n.d.). What is a Decision Tree. Retrieved March 12, 2023, from https://www.ibm.com/topics/decision-trees

[11] IBM. (n.d.). What is the k-nearest neighbors algorithm? Retrieved March 12, 2023, from https://www.ibm.com/topics/knn

[12] scikit-learn. (n.d.). Support Vector Machines. Retrieved March 13, 2023, from https://scikit-learn.org/stable/modules/svm.html

[13] Real Python. (n.d.). Logistic Regression in Python. Retrieved March 13, 2023, from https://realpython.com/logistic-regression-python/

Appendix

Rockify Source Code:

https://github.com/open-minded13/2023_Winter_Managing_Data_and_Signal_Processing_at_UW/tree/main/11.%20Final%20Project%20-%20Rockify