Classification of Fall Out Boy Eras

Shifra L. Isaacs, Joseph Yudelson, Dr. Endre Boros

Abstract

This paper explored the use of machine learning techniques to differentiate between two different musical eras of the same rock band, including the technique of Logistic Regression.

Logistic regression (LR) is a widely-used statistical modeling method for binary classification in supervised machine learning. It is often used to predict whether a given event belongs to one of two categories. The process helps data scientists understand which variables are good predictors of class membership. Applications of logistic regression include loan classification in the financial industry and predicting susceptibility to disease in the medical field.

In this particular project, a dataset was constructed using data from Spotify and Genius consisting of songs and lyrics written by the band Fall Out Boy. A logistic regression model was developed from scratch to classify the songs and lyrics into one of two eras of the band: before their 2009 hiatus and afterward. The study aimed to determine if a computer could differentiate between the two eras. The model was also tested against other binary classification algorithms, including Random Forest and Support Vector Machines.

2 Data Extraction

Audio and lyric features from Fall Out Boy's (FOB) music were extracted utilizing the Spotipy and LyricsGenius APIs. Detailed information about the songs used for the project can be found in Appendix B of the supplemental document. The extracted data was subsequently cleaned and combined to form a cohesive dataset, which was then explored and visualized. Before modeling, the dataset was split into a training set of 98 records and test set of 43 records, for a total of 141 records.

The data extraction stage presented several challenges, such as the need to clean up sub-headings within the lyrics using complex regular expressions. Additionally, the Spotipy API featured interesting edge cases, such as a post-hiatus Elton John cover that was incorrectly labeled as a 1973 release.

Additionally, one Spotify audio feature proved difficult to handle due to its format. For example, the API described song durations in milliseconds, which were subsequently converted to minutes. Care was taken to leave comments explaining that a value of 3.05, for example, should be read as 3.05 minutes rather than 3 minutes and 5 seconds.

Descriptions for all the Spotify features can be found in Appendix A of the supplemental document.

2 Data Analysis

Prior to the modeling process, it is common practice to engage in exploratory data analysis (EDA) and dive deeper into the features that will be modeled. This section will discuss the analyses performed on the Fall Out Boy songs dataset.

For the quantitative audio features, histograms were created to provide a graphical representation of the data distributions. In addition, scatterplot matrices were used to visualize any correlations between the different features.

Figure 1 shows the distributions of the valence and tempo features and indicates the possibility of a small positive correlation between them. This is feasible considering that happier, high-valence songs, broadly tend to have faster tempos.

To analyze the categorical audio features, the distribution of categories was examined across the pre-hiatus and post-hiatus classes.

As for the lyrical features, a hypothesis was formed that pre-hiatus songs generally contain more unique words than post-hiatus songs. A preliminary analysis of the relevant histograms suggests that this may be the case.

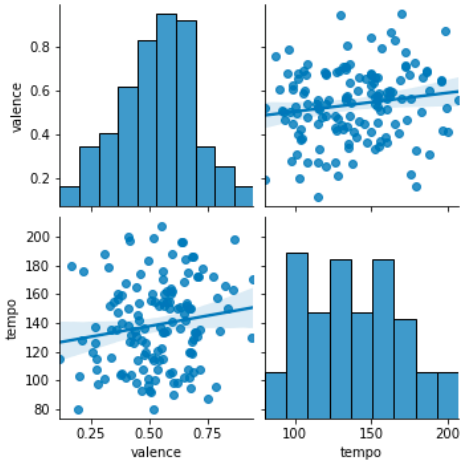

FIGURE 1: *Scatterplot matrix of valence and tempo audio features. Valence is the mood of a song, with high values indicating positive mood.*

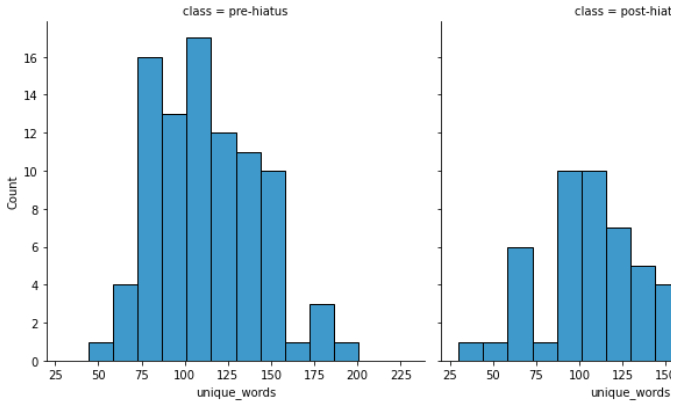

FIGURE 2: *Distribution of unique_words feature across pre-hiatus and post-hiatus songs*

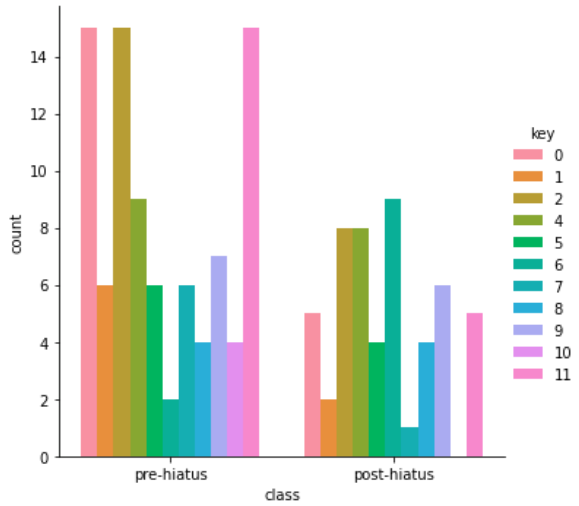

FIGURE 3: *Frequency Plot for Key Feature Across Classes*

However, a two-sample z-test of these datasets yielded a p-value of approximately 0.46. A z-test is a statistical hypothesis test used to compare the means of two independent samples and determine if they are significantly different.

The p-value of 0.46 suggests that there is not enough evidence to reject the null hypothesis, which states that the two samples have no significant difference. This indicates no evidence of a substantial difference in the mean unique lyrics between pre-hiatus songs and post-hiatus songs.

3 FEATURE ENGINEERING

In machine learning modeling, it is common practice to scale all the numerical features within a fixed range while preserving their relative distance. This technique speeds up the process and improves model accuracy because algorithms are designed to work with a fixed value set.

The majority of audio features obtained from the Spotipy API were already scaled between 0 and 1; therefore, MinMaxScaler was used to scale other variables to the same range. Categorical features and the target variable were then one-hot encoded.

Backward feature selection was employed to determine the optimal feature set based on F1 scores, which measure a model's ability to predict true positives accurately and identify true positives correctly. The algorithm resulted in the selection of 8 features: danceability, instrumentalness, total_words, mode, key_6, key_9, key_10, and key_11.

The mode feature, which indicates whether a song's key is major or minor, proved to be a strong predictor despite its inability to capture the musicality of most songs. The key numbers above refer to specific musical notes; for example, key_6 refers to the musical key of F. Meanwhile, the instrumentalness feature represents the likelihood from 0-1 that the track is instrumental and contains no vocals.

After a heatmap correlation matrix confirmed that there were no significant linear correlations among the selected features, the modeling step was initiated.

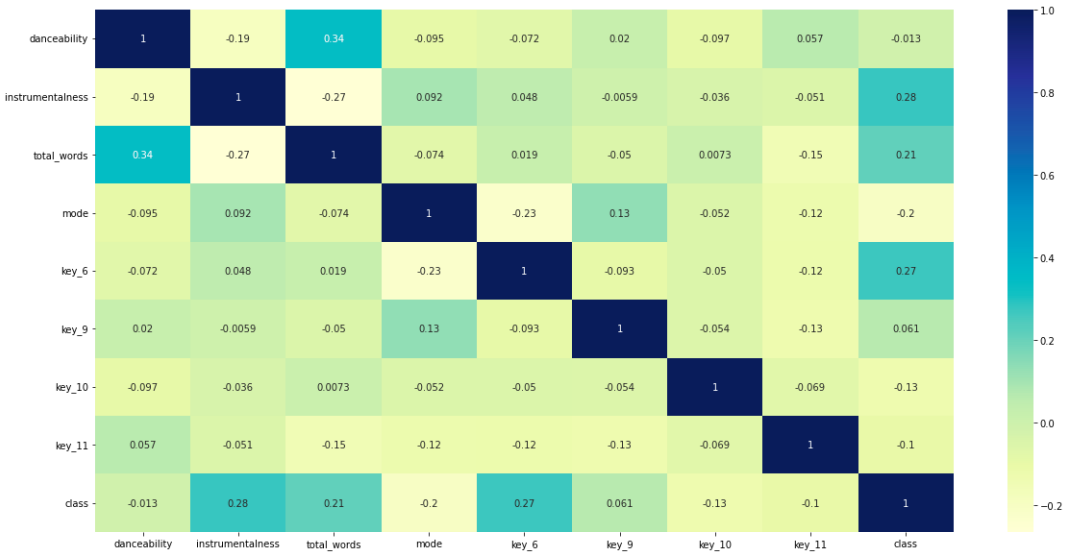

FIGURE 4: *Heatmap Showing Minimal Correlation in Feature Set*

4 Logistic Regression From Scratch

A Logistic Regression (LR) algorithm was constructed from scratch in Python, along with class methods to facilitate model evaluation and visualization. The class was well-documented in section VI of the supplemental document.

Before describing the process in detail, this section will introduce the general modeling process:

- Model selection: Design modeling task based on the item to be modeled or predicted and select the model best suited to that task
- Data pre-processing: Prepare data for modeling and split data into training and testing sets
    - Typically 80% of the data is used for training and 20% for testing
    - Sometimes a third validation set is used, but this dataset is far too small for that
- Model building
- Model training: Run the model iteratively on the training set and update the model parameters on each epoch to improve the model results
- Model testing: Run the model on the test data on each epoch
- Model evaluation: Run metrics such as accuracy and F1 Score to determine quality of model performance

After creating the basic class properties and methods for the model, the focus was on the algorithm itself. A stable sigmoid function was chosen instead of a typical sigmoid function; the regular sigmoid can sometimes result in an impossible denominator of zero, whereas the changes in the stable sigmoid prevent zero-division errors.

The most complex aspect of the algorithm was the implementation of the fit method, which involved several steps:

- collecting properties from the input data.
- constructing a linear model based on the input data.
- computing binary cross-entropy loss as a measure of model error.
- performing gradient descent during training.
- storing loss values at each training and testing stage.

The final steps of the algorithm involved converting the fit scores to probabilities using the stable sigmoid function, and then converting those probabilities into class predictions for either the pre-hiatus or post-hiatus era.

To evaluate the model, a method was added to calculate a confusion matrix and conventional performance metrics, including accuracy, precision, recall, and the F1 score.

A model tracking method was also included to graph training and test error.

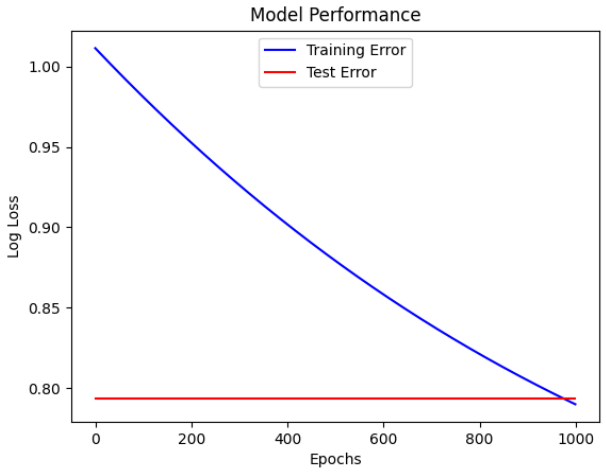

FIGURE 5: *Graph of Training and Test Error*

Figure 5 shows that the training error declined gradually over the training period. Meanwhile, the testing error remained at a plateau near 0.78 throughout training. This is quite unusual; normally the testing error should decline as well. This anomaly was not investigated further because the model performed decently well despite it.

However, the lack of change is likely due to the small size of the test sample, which was only 43 records. It's also possible that the model was slightly overfitting, or fitting too well, to the training data such that the training process didn't significantly impact the testing performance.

As a final step, a naive model was included for comparison purposes that classified FOB songs randomly.

5 COMPARATIVE MODELING RESULTS

To contextualize the performance of the custom-built logistic regression model, several other binary classification algorithms were run using the SciKit Learn Python library.

As an additional experiment, logistic regression was implemented as a single-layer neural network (NN) using a linear layer that was optimized with stochastic gradient descent. These complex mathematical processes were implemented using just two concise lines of code in PyTorch.

The model hyperparameters were tuned by implementing a grid search cross-validation algorithm. The model correctness was then measured by counting the numbers of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) found by the model. Those counts are expanded into specific metrics which are used to evaluate model performance:

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

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

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

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

FIGURE 6: *Comparative Model Metrics (SOURCE)*

The resulting metrics from all models were as follows:

Model	Accuracy	F1 Score	Precision	Recall
SVC	62.79%	52.94%	52.94%	52.94%
LinearSVC	67.44%	36.36%	23.53%	80.00%
GaussianNB	69.77%	60.61%	58.82%	62.50%
KNN	65.12%	44.44%	35.29%	60.00%
NN	25.58%	10.53%	100.00%	11.76%
Random Forest	79.07%	70.97%	64.71%	78.57%
LR (Scratch)	41.86%	28.81%	40.48%	100.00%
LR (SciKit)	76.74%	64.29%	52.94%	81.82%

FIGURE 7: *Comparative model metrics. SVC refers to Support Vector Machines, GaussianNB refers to Gaussian Naive Bayes, and KNN refers to K-Nearest Neighbors*

As expected, the Random Forest Classifier performed well on all metrics, as it is an ensemble method that utilizes multiple learners. Both implementations of the logistic regression algorithm performed better than Random Forest in terms of recall, which initially suggests impressive performance. However, the low accuracy of the custom-built logistic regression model renders the high recall essentially meaningless.

It is worth noting that the optimal logistic regression algorithm using the SciKit Learn library employed L1 regularization, which is typically used to prevent overfitting. Although the conventional logistic regression model was not overfitting and did not appear to require L1 regularization, its inclusion improved all of the recorded metrics.

6 Conclusion and Next Steps

This project involved several key steps:

- Aggregating data from multiple APIs.
- Exploring and analyzing the sourced data for feature engineering.
- Building models to evaluate their performance on a binary classification task.
- Answering the question: Can a machine differentiate between the old and new music of Fall Out Boy (FOB)?

The details of all of these steps can be found in the supplemental document along with relevant code blocks and information sources.

The results of this project indicate that logistic regression performed surprisingly well on a small dataset of around 140 rows. Random Forest also performed well, as the ensemble learning compensated for the smaller size of the data.

Overall, this project shows that rudimentary AI can still simulate the human experience to some degree. It's quite a human thing to notice differences in musical styles, or to differentiate an artist's old sound from their new one, and this simple AI can still accomplish that. It's noteworthy that complex generative models like Chat GPT and DALL E 2 specialize in tasks such as writing and visual art. We once thought machines would struggle with these creative tasks, but in the current generative AI landscape, it seems that tasks thought to be most human are arguably the most conducive to AI.

A variety of methods are available to explore these AI tasks. An alternative approach for this project would have been to analyze FOB songs as complex time series, using deep learning to extract relevant features, and fine-tuning a complex neural network as precisely as possible. However, the goal of this project was to gain experience with data wrangling and popular machine learning algorithms, as opposed to advanced time series modeling and deep learning.

The next steps for this project would include further model tuning and optimization for the algorithms discussed in the comparative modeling stage. Next steps could also include the alternative approach of rigorous time series analysis. This approach would be helpful because researchers could derive their own numerical audio features using time series patterns instead of using Spotify's limited features. This process would require specialized audio analysis and time series tools which could be found in the StatsModels, Tensorflow-IO, and Librosa Python packages■

7 Acknowledgments

I would like to express my appreciation to the following people who contributed a great deal to this project:

Professor Endre Boros, for supervising this project, advocating on my behalf on a number of occasions, and going above and beyond to give me the best possible college experience.

Fall Out Boy, for their wonderful music and for appreciating this project.

Joey Yudelson, for providing strategic recommendations, deep Python expertise, and impeccable technical editing.

Nick Singh, my mentor, for encouraging me to work on data science portfolio projects and teaching me how to best leverage my results.

Siddhant Kochrekar, for providing advice concerning the feature engineering process, model tuning, and out-of-the-box ideas to improve this project.

Vastava, for inspiring this project with her content and her code.

Zack Ovits, for helping me build the project directory, and for providing consistent advice on best practices throughout the duration of this project.

8 References

[1] Brownlee, J. (2020, September 14). *Hyperparameter Optimization With Random Search and Grid Search*. Machine Learning Mastery. Retrieved November 11, 1111, from *machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/*

[2] [Coding Lane]. (2021, February 4). *Logistic Regression in Python from Scratch | Simply Explained* [Video]. YouTube. *youtube.com/watch?v=nzNp05AyBM8*

[3] Dola, P. (2020, September 9). *Exploratory Analysis of Spotify Tracks: How has music changed over the past 100 years?* RPubs. Retrieved January 1, 1111, from *rpubs.com/PeterDola/SpotifyTracks*

[4] [Elbert]. (2021, January 5). *How to automate with Python, Spotipy, and LyricsGenius* [Video]. YouTube. *youtube.com/watch?v=cU8YH2rhN6A*

[5] FOB Song Data. (n.d.). Spotify. *open.spotify.com/playlist/0ubKnC9ctDVxlGbF5jICvQ?si=d1a259083136490a&nd=1*

[6] Ho1yShif. (n.d.). *GitHub - Ho1yShif/FOB_LR_Public: Independent Study Project: Classification of Fall Out Boy eras*. GitHub. *github.com/Ho1yShif/FOB_LR_Public*

[7] Klosterman, S. (2019). *Data Science Projects with Python*. Packt.

[8] Loeber P. (2019, September 15). *Logistic Regression in Python - Machine Learning From Scratch 03 - Python Tutorial* [Video]. YouTube. *youtube.com/watch?v=JDU3AzH3WKg*

[9] Loeber P. (2019, December 30). *PyTorch Tutorial 08 - Logistic Regression* [Video]. YouTube. *youtube.com/watch?v=OGpQxIkR4ao*

[10] Parveez, S., & Iriondo, R. (2020, December 28). *Gradient Descent for Machine Learning (ML) 101 with Python Tutorial*. Towards AI. Retrieved November 11, 1111, from *pub.towardsai.net/gradient-descent-algorithm-for-machine-learning-python-tutorial-ml-9ded189ec556*

[11] Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python

[12] Scikit-learn developers (2021). 1.1.11. Logistic regression. Scikit-learn. *scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html*

[13] Scikit-learn developers (2021). 1.1.2. RandomForestClassifier. Scikit-learn. *scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html*

[14] Scikit-learn developers (2021). 1.4. Support Vector Machines. Scikit-learn. *scikit-learn.org/stable/modules/svm.html*

[15] Scikit-learn developers (2021). 1.9. Naive Bayes. Scikit-learn. *scikit-learn.org/stable/modules/naive_bayes.html*

[16] Statistics Solutions (2021). Assumptions of Logistic Regression. *statisticssolutions.com/free-resources/directory-of-statistical-analyses/assumptions-of-logistic-regression/*

[17] [StatQuest With Josh Starmer]. (2018, June 4). *Logistic Regression Details Pt1: Coefficients* [Video]. YouTube. *youtube.com/watch?v=vN5cNN2-HWE*

[18] [StatQuest With Josh Starmer]. (2018, June 11). *Logistic Regression Details Pt 2: Maximum Likelihood* [Video]. YouTube. *youtube.com/watch?v=BfKanl1aSG0*

[19] [StatQuest With Josh Starmer]. (2018, June 18). *Logistic Regression Details Pt 3: R-squared and p-value* [Video]. YouTube. *youtube.com/watch?v=xxFYro8QuXA*

[20] [StatQuest With Josh Starmer]. (2018, May 7). *Odds and Log(Odds), Clearly Explained!!!* [Video]. YouTube. *youtube.com/watch?v=ARfXDSkQf1Y*

[21] [StatQuest With Josh Starmer]. (2018, March 5). *StatQuest: Logistic Regression* [Video]. YouTube. *youtube.com/watch?v=yIYKR4sgzI8*

[22] [Vastava]. (2020, December 4). *I trained an AI to tell me CORPSE's music genre | data science* [Video]. YouTube. *youtube.com/watch?v=VTU6JLa70VY*

[23] Vastava, S. (2021, May 25). Spotify Genre Classifier [Repository]. GitHub. *github.com/vastava/data-science-projects/blob/master/spotify-genre-classifier/get-spotify-data.ipynb*

[24] Z. (2020, October 13). *The 6 Assumptions of Logistic Regression (With Examples)*. Statology. Retrieved November 11, 1111, from *statology.org/assumptions-of-logistic-regression/*

Shifra Isaacs is a data scientist, data analyst, and technical writer with industry experience at Annalect, JPMorgan Chase, CrashCourse, DataLemur, and more. Shifra's passion for data extends beyond her professional work. She has contributed to the field through publications, projects, and educational content that makes data and programming more accessible.

With a diverse skill set including Python, SQL, BI tools, documentation, and product management, Shifra demonstrates versatility and adaptability in addressing complex challenges. Her track record of delivering impactful solutions, coupled with her strong analytical prowess and business acumen, establishes Shifra as a valuable asset in the field of data science.