Predicting Song Success using Spotify Data

Lars L. Ankile, Nora Hallqvist, Ron Nachum, Sebastian Weisshaar

Harvard AC 209A: Data Science I

Harvard School of Engineering and Applied Sciences

November 2022

Abstract

The project aims to predict whether a song is a hit or flop based on song metrics provided by Spotify. The used data consists of 44,106 tracks across six decades (the 60s - 2010s), collected from Kaggle. Six models were trained on the data set, of which the stacking model achieved the best test accuracy of 0.8055. FPR rates and feature importance were analyzed, and we find that predictors of success have been stable over the decades. Further work could investigate the use of features containing more predictive power to overcome the 80% accuracy threshold we found.

1 Project Question

The aim of the project is to answer the following question:

How well can one predict whether a song is a hit or flop based on the song metrics provided by Spotify?

We will build models based on a dataset with 16 features to predict if a song is a hit or flop. The 16 features included in the model are `danceability`, `energy`, `key`, `loudness`, `mode`, `speechiness`, `acousticness`, `instrumentalness`, `liveness`, `valence`, `tempo`, `duration_ms`, `time_signature`, `chorus_hit`, `sections`, and `decade`. See Subsection 2.1 and Subsection 2.2 for an in-depth exploration of the data.

2 Methods & Models

2.1 Dataset and Features

The data set consists of 44,106 tracks from six decades (the 60s - 2010s) and was collected from Kaggle [1]. It consists of 20 features, where the response variable is either 1 (Hit) or 0 (Flop), indicating whether the track is featured on the Billboard Hot 100 songs. The features `track`, `uri`, and `orig_idx` are unique to the song and hence are not considered helpful in predicting a song's popularity. In addition, we do not include the variable `artist` to remove the influence of an artist's popularity (a feature external to the actual song itself) on the track's performance. The remaining 16 features Figure 1 will be used to build a model.

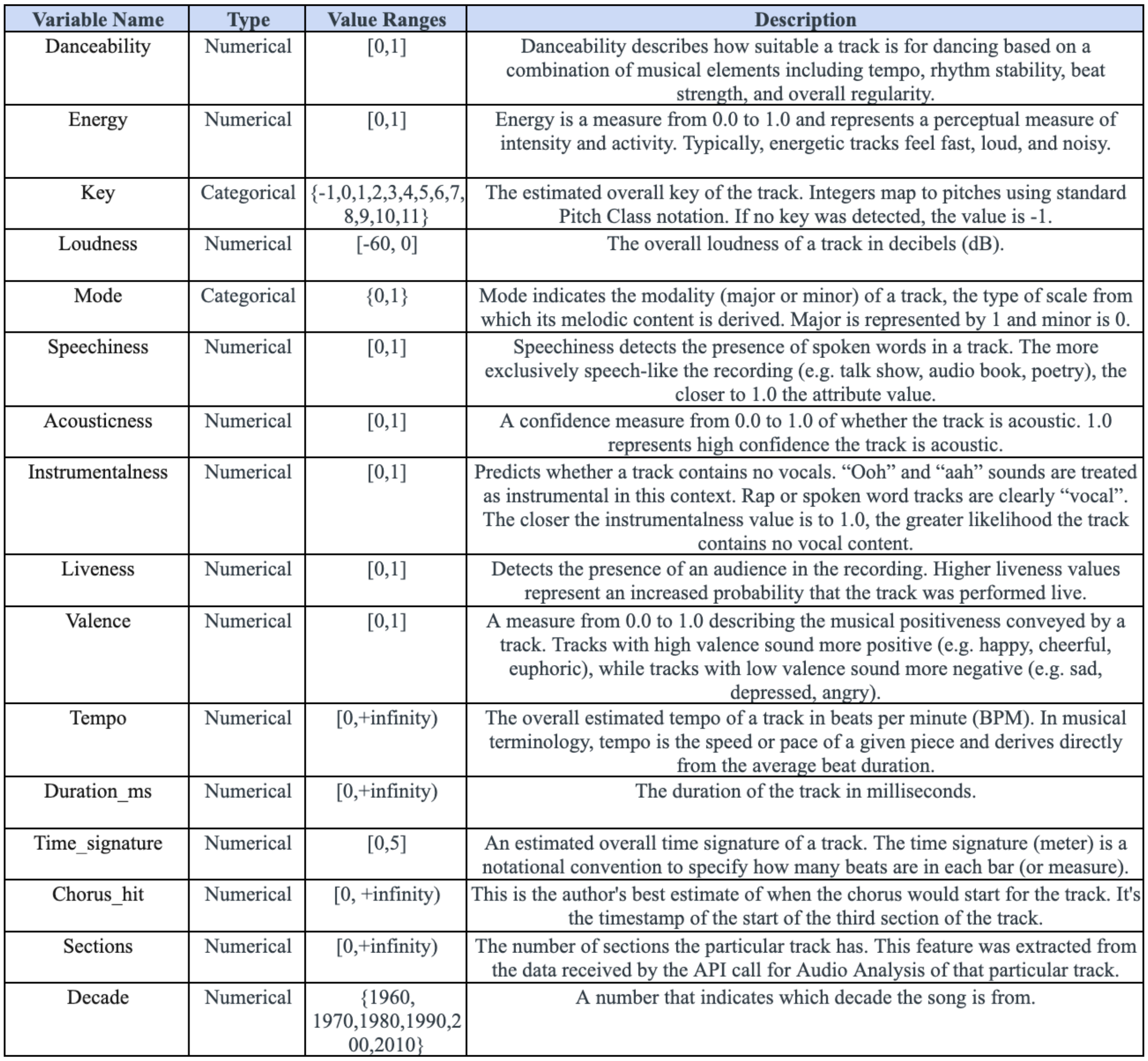
After analyzing the type of features included in the data set along with their relevance in predicting the target, we explored missing values. No NA or null entries in the data set were discovered. Lastly, the data set was split into a train and test set with 80% training data and 20% testing data, using a random state of 109. The test data is stored in a `test.csv` file for performance evaluation.

2.2 Exploratory Data Analysis

To better understand the meaning of the features, we analyzed three songs. The three songs are "We are never ever getting back together" by Taylor Swift, "Dancing Queen" by ABBA, and "Boom Boom Pow" by Black Eyed Peas. The songs were selected to be diverse in their music profile. Figure 2 shows the scaled variables for all three songs.

We see that the most significant difference is in the liveness of the songs. "Boom Boom Pow" has the highest liveness, whereas Taylor Swift has less than average liveness. We invite the reader to validate this result by listening to the songs. Interestingly all songs are less than average in tempo, which is unsurprising when listening to the tracks. We suspect this is caused by songs with a very high tempo being included in the data set. Therefore perceived fast songs might not have a high tempo compared to these other songs.

Figure 3 shows that the response variable is evenly distributed between 'Hit' and 'Flop' in all decades. This is useful as we do not have to account for class imbalances in our models.



Variable Name	Type	Value Ranges	Description
Danceability	Numerical	[0,1]	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity.
Energy	Numerical	[0,1]	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.
Key	Categorical	{-1,0,1,2,3,4,5,6,7,8,9,10,11}	The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation. If no key was detected, the value is -1.
Loudness	Numerical	[-60, 0]	The overall loudness of a track in decibels (dB).
Mode	Categorical	{0,1}	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
Speechiness	Numerical	[0,1]	Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.
Acousticness	Numerical	[0,1]	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
Instrumentalness	Numerical	[0,1]	Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly “vocal”. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content.
Liveness	Numerical	[0,1]	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.
Valence	Numerical	[0,1]	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
Tempo	Numerical	[0,+infinity)	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
Duration_ms	Numerical	[0,+infinity)	The duration of the track in milliseconds.
Time_signature	Numerical	[0,5]	An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).
Chorus_hit	Numerical	[0, +infinity)	This is the author's best estimate of when the chorus would start for the track. It's the timestamp of the start of the third section of the track.
Sections	Numerical	[0,+infinity)	The number of sections the particular track has. This feature was extracted from the data received by the API call for Audio Analysis of that particular track.
Decade	Numerical	{1960, 1970,1980,1990,200,2010}	A number that indicates which decade the song is from.

Figure 1: Summary of features used in modeling.

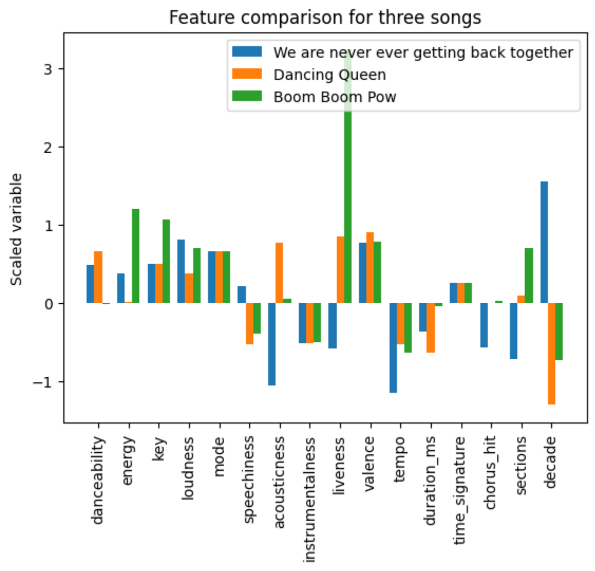


Figure 2: Feature comparison for three songs.

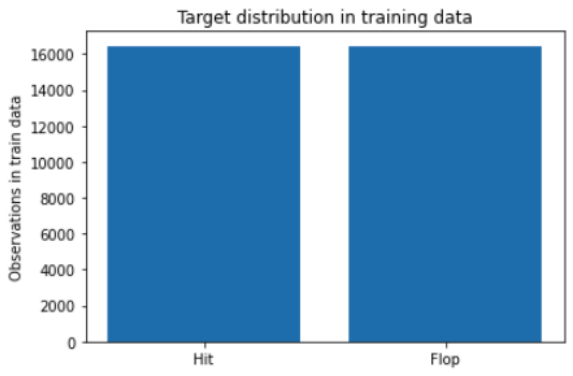


Figure 3: The number of samples in each category for our data. The classes are perfectly balanced, with is helpful for downstream modeling.

Decade	Target Mean
1960	0.50
1970	0.50
1980	0.50
1990	0.50
2000	0.51
2010	0.50

Table 1: Training Data Target Distribution

Based on Figure 4, `danceability`, `instrumentalness`,`energy`, `acousticness`, and `valence` seem to have the most potential to predict a `hit` or a `flop`. This can be seen by a significant difference in these variables between `hits` and `flops`. For example, tracks labeled as hit have a mean danceability of 0.6019, while flop tracks have a mean danceability of 0.4795.

From the correlation table in Figure 5, we can observe that duration (ms) and sections have the highest correlation of 0.89, followed by energy and loudness with a correlation of 0.77. In addition, acousticness is negatively correlated with both loudness and energy, with values of -0.57 and -0.71, respectively. This makes intuitive sense since the more a song skews to being energetic, the less it becomes acoustic. Our response variable target is weakly correlated with all features, which might be problematic in modeling. The strongest correlation of -0.41 is observed between target and instrumentalness, closely followed by target and danceability with 0.34. This seems to imply that no individual variable has strong predictive value independently, but that combinations of them could yield useful information.

We plotted four feature frequencies over time in Figure 5. Most notably, we can observe that average loudness has decreased over the decades until 2020, and there has been an overall increase in energy and speechiness. An increase in energy might be expected with the arrival of electronic dance music. This may potentially be related to the growing mainstream popularity of hip-hop and rap music, though we do not have a solid explanation for the rise in speechiness. On the contrary, current songs are generally perceived as less vocal overall.

2.3 The Baseline Model

The training data set has a balanced distribution in the target variable. Therefore, any model we train should be able to perform better than a coin flip guess. We used logistic regression as a baseline model. Using cross-validation with 10 folds, we found a mean accuracy on the validation set of 0.55. This is just slightly better than a random guess.

Furthermore, we trained a single tree classifier with a depth of 10 and a bagging method on the training data to gauge the potential score of different models. In no model have we performed hyperparameter tuning yet; instead, we used them to get an initial grasp of various model performances. The results of this exploration can be found in Table 2. Compared with the base logistic regression model, the models perform very well. This is a hint that further model improvement could result in accuracy scores of around 80%.

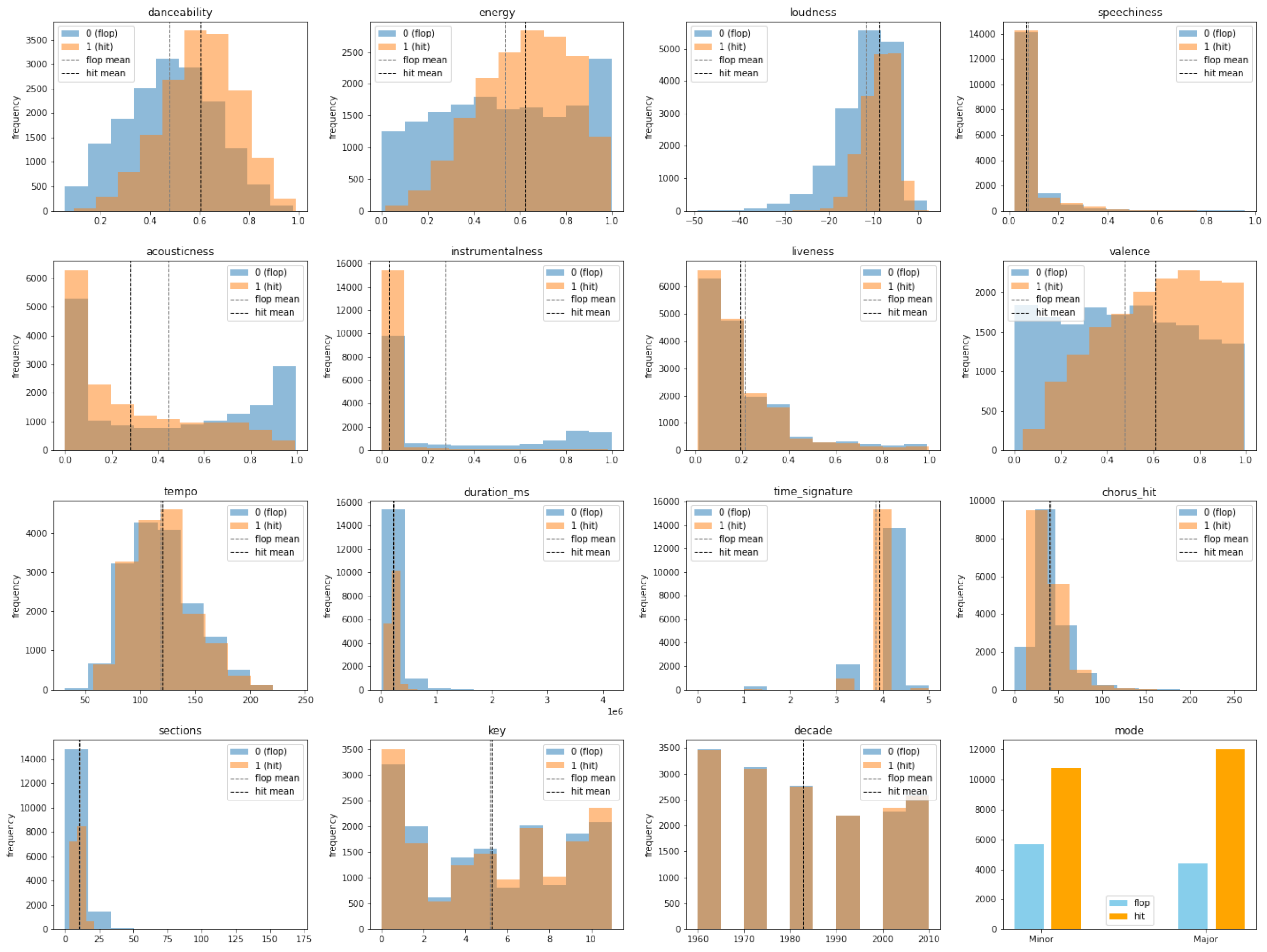


Figure 4: Marginal distributions of the predictors by track *hit* and *flop* in the training data.

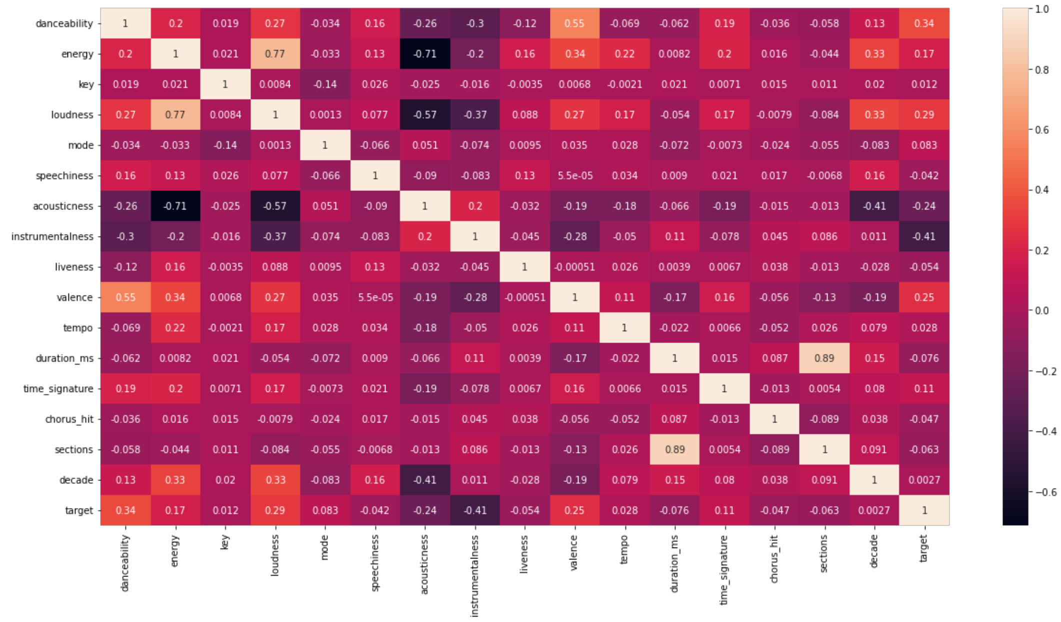


Figure 5: Features frequency over time; Correlation plot predictors.

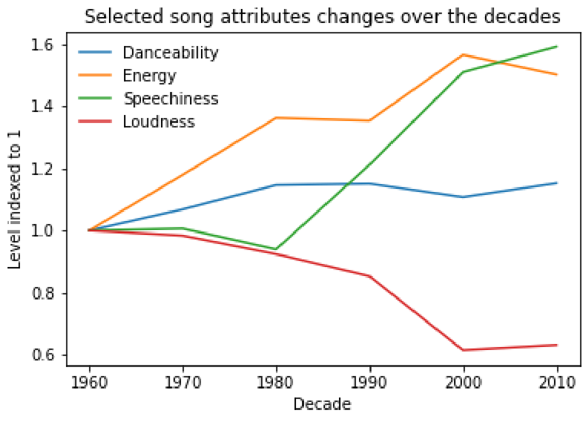


Figure 6: Average of song danceability, energy, speechiness, loudness grouped by decade.

Model	**CV Mean**	**CV σ**
Naive Predictor	0.4997	0
Logistic Regression	0.5481	0.0465
Single Tree	0.7598	**0.0054**
Bagging Tree	**0.7924**	0.0061

Table 2: The Cross-Validation (CV) results of the baseline models serve as an indication of the benchmark accuracy.

2.4 Model Descriptions

To predict a song's success, we used 6 different machine-learning algorithms:

- Single Decision Tree, Logistic Regression, Random Forests, AdaBoost, Neural Networks, Stacking

We chose the above methods for multiple reasons. For one, considering the binary nature of the response variable, the data fits well for logistic regression. Secondly, recent research shows that tree-based models are superior to neural models in many instances for tabular data [2]. Lastly, our final goal was a well-fitted ensemble method. We worked our way backward to achieve this by asking what type of models would form good building blocks for our ensemble methods.

Firstly, we built a single Decision Tree classifier and a Logistic Regression. Due to many features in the data set, we have decided to use penalized Logistic Regression. Specifically, we will perform Ridge regularization (i.e $l2$), a technique that lowers the dimension of the feature space by shrinking the coefficients of less essential features toward zero. As required when applying regularization, the data set is scaled. This is necessary to avoid features with large units of measurement being unfairly penalized. The scaled train data set is used to train all models for consistency.

Furthermore, we trained more complicated methods like Random Forest and AdaBoost. These models were then combined in a stacking ensemble classifier. Finally, we also trained a four-layer neural network. Below we briefly describe the individual models and the stacking method.

Single Tree Classifier is used to predict a qualitative response by binary recursive partitioning. This is an iterative process of splitting the data into partitions until you isolate the data points in each class. When building the single Decision Tree, we use the 'Gini Index,' a node purity measure, to evaluate each split's quality. By cross-validation, we tune the single hyper-parameter, tree depth.

Logistic Regression models the probability of an event belonging to a specific category by having the log-odds of that event be a linear combination of the predictors, i.e.,

$$\log\left[\frac{p(x)}{1-p(x)}\right] = \beta X$$

In this project, we use binary logistic regression where 0 and 1 represent 'flop' and 'hit,' respectively.

Random Forests is an ensemble learning method for classification that is an evolution of bagging. It only considers a subset of predictors at each split and decorrelates the different classification trees. With the use of `GridSearchCV`, we obtain the max depth and number of estimator that maximizes the validation accuracy. We keep the number of features considered at each split constant at 4 (i.e., `max_features="sqrt"`)

AdaBoost, short for Adaptive Boosting, is an ensemble model where each subsequent classifier is grown from previous classifiers by weighting observations that the previous learner incorrectly classified higher. In contrast to Random Forests, the principle of AdaBoost is to train a series of 'weak' learners, i.e., decision trees with small max depth. With the help of `GridSearchCV`, we find the optimal set of hyper-parameters, including tree depth, number of estimators, and learning rate. However, before `GridSearchCV` is performed, we determine feasible parameter spaces for each hyper-parameter by tuning on a single validation set. In the end, 400 models were trained with max depth between 1-4, a learning rate between 10^{-4}-1, and the number of estimators varying between 100-600. The algorithm 'SAMME.R' was held constant.

Neural Networks (NN) are a complex space of models, and we will only briefly describe them here. Please refer to works like [3, 4] for a more in-depth look at the technical details. In short, NNs approximate complex functions by combining several successive matrix multiplications with so-called non-linearities, i.e., non-linear functions. Concretely, one layer of an NN can be described by the equation $L_i(\boldsymbol{X}) = g(\boldsymbol{W}_i^T + \boldsymbol{b}_i)$, where $\boldsymbol{X}$ is the input data or the output of the previous layer, $\boldsymbol{W}$ is the model parameters or weights, $\boldsymbol{b}$ is the bias or offset term, and g is the non-linearity, often the Rectified Linear Unit `ReLU` $\max\{0, x\}$ for intermediate layers or the sigmoid $\frac{1}{1+e^{-x}}$ for the output layer in the binary classification case. A full NN model would chain an arbitrary number of these layers together $L_1 \circ L_2 \circ ... \circ L_L$ to make the resulting function approximator as complex as desired. One needs the non-linearity g in this equation because without it, all layers could trivially be represented by a single matrix $\boldsymbol{W}' = \boldsymbol{W}_1 \boldsymbol{W}_2 ... \boldsymbol{W}_L$, i.e., the whole model is reduced to a linear transformation.

One drawback with NNs is that they are infamously sensitive to the choice of hyper-parameters and can often take a long time to train. In our case, however, since the data set is sufficiently small, we could perform a hyper-parameter search with a grid search in a manageable time frame. As our parameters in the search, we considered models with 2 to 4 hidden layers, 8 to 32 neurons at each layer, a range of learning rates, and different dropout and weight decay levels for regularization. The final model has 3 layers of 32, 16, and 16 neurons. Furthermore, it uses the Adam optimizer with a learning rate of 0.01 and has 20% dropout and no weight decay.

Stacked Generalization is an ensemble algorithm that combines several different prediction models into a single model. The method aims to combine the strengths of individual classifiers into one single classifier. Besides majority voting, this ensemble method trains an extra model on top of the base models. The second-level model uses the predictions from the base models as variables. This allows the stacked model to identify which base models are useful for which data points.

3 Results

To evaluate the model performance, we have concentrated on accuracy scores. Investing in a track or artist is very costly, and record labels are interested in avoiding false positive predictions (FPR) as this would entail investing in songs that flop. Therefore, we also report the confusion matrix of each model to be able to analyze this aspect as well.

3.1 Overall Results

In this section, we present the final accuracy scores on the train and test set for all models after hyperparameter tuning. There are a couple of notable findings here. First, the Stacking model performed the best. Second, Stacking and AdaBoost outperformed the Neural Network (NN). Third, all models achieved good results, and the gap between the best and worst is not large ($\sim 5.2\%$).

Model	Train	Test	FPR
Single Decision Tree	0.7902	0.7637	0.3263
Logistic regression	0.7915	0.7889	0.2749
Random Forest	0.9696	0.8055	0.2566
AdaBoost	0.8036	0.8035	0.2566
Neural Network	0.8073	0.7985	0.2527
Stacking	**0.9617**	**0.8069**	**0.2320**

Table 3: Accuracy results on the train and test set, and False-Positive Rate (FPR) on the test set, for all fine-tuned models.

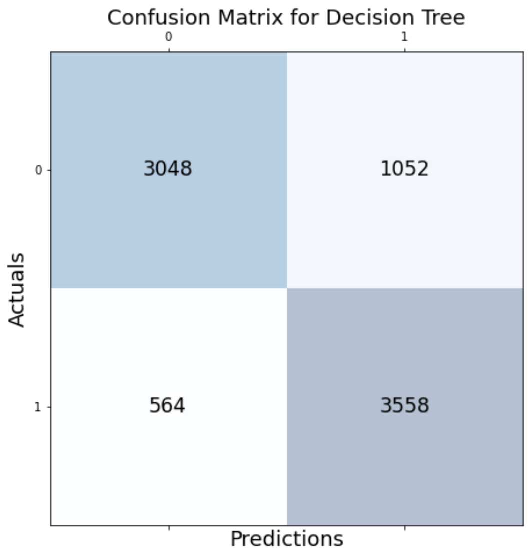


Figure 7: Confusion matrix for the Single Tree Classifier.

3.2 Single Decision Tree

The single decision tree is not surprisingly the worst-performing out of all models, with the lowest test accuracy of 76.4%. In Figure 7, we can see that the FPR is 32.68% which is the highest rate amongst all other models.

3.3 Logistic Regression

The logistic regression model performs better than the single decision tree, with a test accuracy of 78.9% and an FPR of 27.49%. Compared to AdaBoost and Stacking, we are somewhat surprised by the performance of logistic regression. This finding could indicate that our dataset is quite easily classifiable and that we are squeezing out more or less all the available signals when we cross the 80% accuracy threshold. Even though logistic regression is a reasonably simple model, it took a considerable amount of computing to train it fully. One reason could be that the normal closed-form solution can not be used for logistic regression and uses numerical solvers instead.

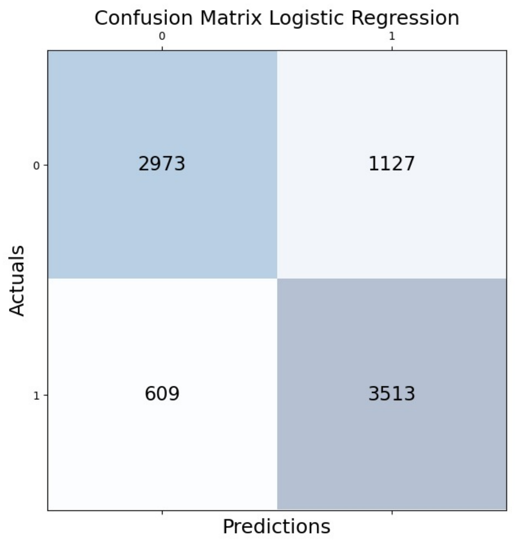


Figure 8: Confusion matrix for the Logistic Regression Model

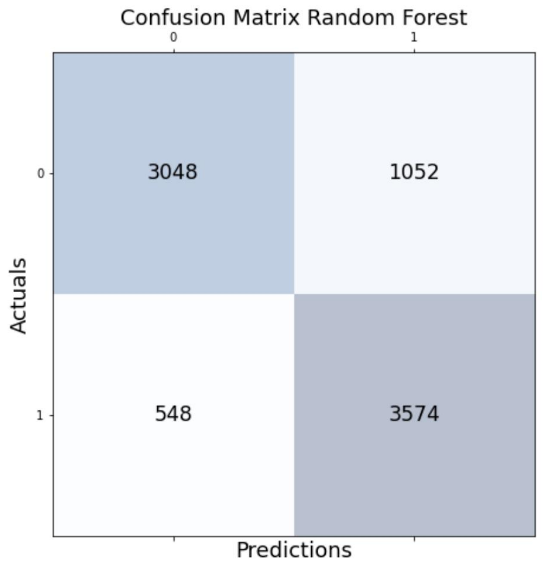


Figure 9: Confusion matrix for the Random Forest Model

3.4 Random Forest

The Random Forest model performed well at a mean test accuracy of 80.3%, i.e., almost equalling AdaBoost. Not surprisingly, it outperforms a single Decision Tree. This can be partially explained by the fact that Random Forest helps reduce variance by averaging a number of observations. Mathematically this can be justified the fact that if given a set of n independent observation $Y_1, ..., Y_n$ each with variance σ^2, the variance of $\bar{Y}$ is given by $\frac{\sigma^2}{n}$. Hence, if we view the predictions of the individual model as individual observations, combining them reduces variance.

In terms of FPR, though, it performs equally to AdaBoost at an FPR of 25.66% (see Figure 9). This is an interesting finding which deserves more analysis in further work. Furthermore, the training of the Random Forest model is much simpler and faster than for AdaBoost, motivating the use of the Random Forest as a go-to first model to try on new data sets.

3.5 AdaBoost

AdaBoost performed very well and came in at a test accuracy of 80.4%, second only to the Stacking model. The model had a similar result on the training set, indicating no over-fitting. Again, as expected an ensemble model outperforms a single Decision Tree. However, in contrast to Random Forest, Adaboost improves test accuracy by focusing on reducing bias. This is accomplished by combining decision stumps (which have low variance but high bias) and placing higher weights on miss classified training observations.

In Figure 10, we see the false positive rate (FPR) of 25.66%, which is good but not the best among the non-stacking models. AdaBoost was also reasonably easy to work with, but the training took considerable time and required overnight grid search.

3.6 Neural Network

The Neural Network (NN) did not perform as well as recent advances in machine learning would promise. The test accuracy came in at 79.5%, i.e., worse than the Random Forest, AdaBoost, and Stacking. From Figure 12, we can see that it has an FPR of 24.22%, which scores the second-best FPR we have observed.

3.7 Stacking Ensemble Model

Stacking performed the best of all models in terms of both test accuracy and FPR, with a score of 80.55 % and 23.51 %, respectively. High test accuracy and low FPR compared to the other models can be explained by the fact that stacking combines the strength of the individual classifiers (i.e, Single Decision Tree, Logistic Regression and AdaBoost) into a single classifier.

3.8 Feature Importance

We calculated the feature importance using permutations to evaluate how the different features affect our model performance. Figure 11 shows the loss of accuracy when permuting the variable. The three most important features are `instrumentalness`, `acousticness`, and `danceability`. This agrees with our initial findings in Subsection 2.2. `Decade` also has a relatively high feature importance. This is interesting as the target vari-

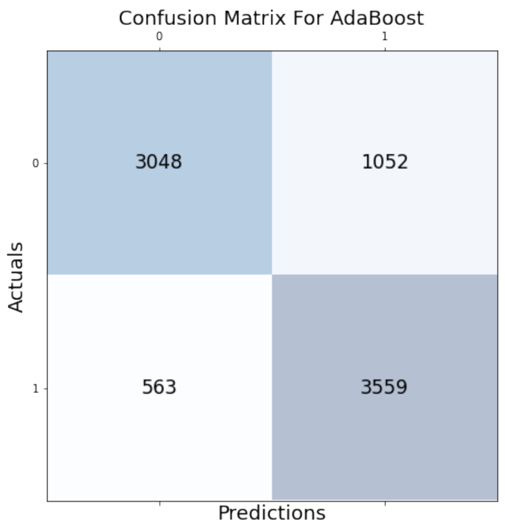


Figure 10: Confusion matrix for the AdaBoost model

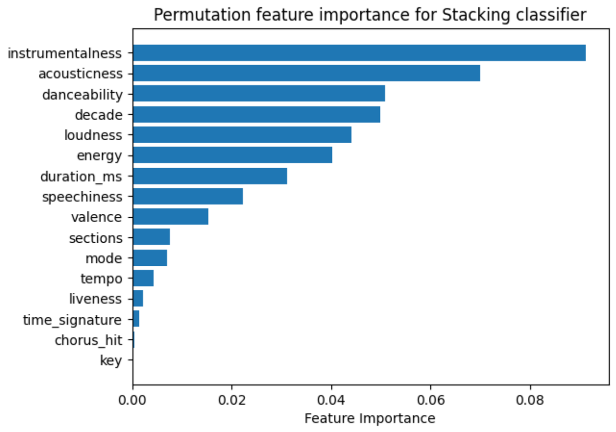


Figure 11: Feature importance in stacking classifier

able is evenly distributed across the decades. A reason for the importance of the decade could be changing preferences. Different features might be important in each decade. Therefore the model needs to know the decade to decide which features to use for the prediction.

To explore this idea further, we analyzed the most important features in each decade. Table 4 shows the top three important features for each decade in the stacking classifier. There seems to be some variation between the decades. The top three consist of almost the same variables in each decade. Therefore we cannot draw any strong conclusions about preferences in each decade. Exploring changing relationships between features and the target variable could be an avenue for future research. It would, for example, be interesting to know what would happen if we released a modern hit back in the 60s or vice versa.

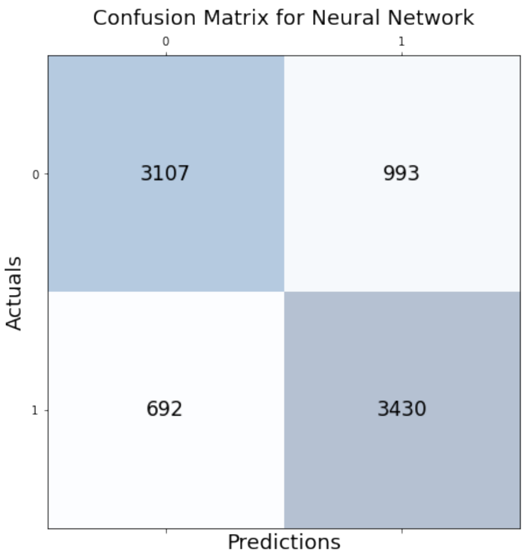


Figure 12: Confusion matrix for the Neural Network

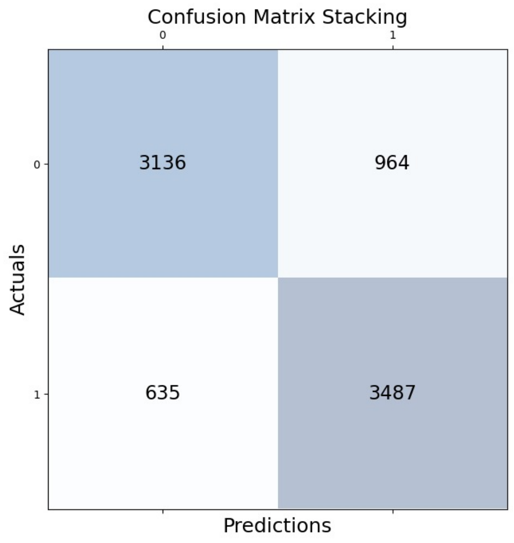


Figure 13: Confusion matrix for the Stacking model

Decade	First	Second	Third
1960	acousticness	energy	instrumentalness
1970	danceability	acousticness	instrumentalness
1980	danceability	acousticness	instrumentalness
1990	instrumentalness	danceability	acousticness
2000	acousticness	danceability	instrumentalness
2010	danceability	acousticness	instrumentalness

Table 4: Most important features over the decades.

3.9 Critical Discussion

While these results show strong predictive power in the features provided by Spotify, there are some limitations to the conclusions.

For one, it is unclear whether all hits are the same. Hits could influence each other and hence make the music converge. Predicting a hit would then be a question of how similar it is to other songs and not how much people will like it.

Second, we did not get to do an exhaustive hyperparameter search over all our models, which means that more performance might be left on the table for our set of models and features. Since the search space of hyperparameters is not a convex space, one would need to conduct a fine-grained search over a large space to have a reasonable chance of finding the optimal parameters. Because of constrained resources (time and computing power), this was not feasible for this project.

4 Conclusion & Discussion

Our results show that we can predict *hits* and *flops* with around 80% accuracy. This finding is somewhat surprising as we would think that the predictors of a song's 'hitness' could not be captured in these simplistic features. We have not used identifiable features (e.g., artist), only features that quantify some aspect of the soundscape. Furthermore, we find that the Stacking Ensemble performs the best in terms of accuracy and FPR. Random Forests, AdaBoost, and the Neural Network have very similar performances on the test data.

Since all models struggle to achieve any higher accuracy than 80%, this could indicate the presence of around 20% irreducible error—meaning that music preference cannot soley be expressed as a function of numbers but sometimes is a question of taste and emotion.

5 Link To Code

`https://datalore.jetbrains.com/notebook/982RhXfLI2QRuMtVKvxPTz/MzJZqPisRWcRWaDwlJah5D/`

References

[1] Kai Middlebrook and Kian Sheik. Song hit prediction: Predicting billboard hits using spotify data. *arXiv preprint arXiv:1908.08609*, 2019.

[2] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[4] Lars Lien Ankile and Kjartan Krange. Exploration of forecasting paradigms and a generalized forecasting framework. Master's thesis, NTNU, 2022.