Spotify Song Hit Prediction
Which song would be the next hit?

Jinglun Gao **Sophy Huang** **Chao Wang** **Hongwen Song**

1 Introduction

Hit Song Science (HSS) has been a popular topic in the field of Music Information Retrieval (MIR), and predicting the success of a given song (hit rate) using the meta data of the song is becoming an increasingly valuable task in the music industry. HSS helps music producer and publisher understand the trends and potentially make more profit by making their works reaching a broader audience. On the other hand, HSS also provides potentials for investors to use AI-assisted tools to understand the music market and make investment accordingly. In this project, we develop machine learning models that predicts the song success rate using The Spotify Hit Predictor Dataset. We also investigate if audio features from Spotify can be considered as determinants of the stream popularity of songs.

2 Related Works

Previous researches on the topic of new product success prediction have identified multiple approaches to investigate the relationship between song data audio features obtained from the Spotify database (e.g. key and tempo) and song popularity, measured by the number of streams that a song has on Spotify. Research from (Lee and Lee, 2018) shows that it is feasible to predict the popularity metrics of a song significantly better than random chance based on its audio signal. Additionally, (Ni et al., 2011) also show that certain audio features such as Loudness, duration and harmonic simplicity correlate with the evolution of musical trends. (Dhanaraj and Logan, 2005) propose features from both songs' lyrics and audio content for prediction of hits and also study a hit detection model based solely on lyrics' features.

Zangerle evaluated a song's likelihood of success based on its acoustic features (Zangerle et al., 2019). In order to jointly exploit low- and high-level features, they used a wide and deep neural network design. They assessed their strategy using the Million Song Dataset, and they defined a song as a hit if it was ever listed in the Billboard Hot 100. According to their research, the suggested strategy can outperform baseline approaches and methods that use either low- or high-level information separately. Their work showed positive results in analyzing potential song success. They similarly used the song's low and high-level features as predictors and used neural networks to achieve 75.04 percent accuracy. In our project, the baseline models have accuracies around 73, so we will expect a better song prediction model in terms of accuracy.

By concentrating on the dance hit song prediction challenge, (Herremans et al., 2014) address the issue of getting an understanding of what constitutes a hit song in their research. The construction of a database of dance hit songs from 1985 to 2013 includes both fundamentally musical elements and more sophisticated elements that capture a time aspect. Dance hit prediction models are constructed and tested using a variety of classifiers. When predicting whether a song will be a "top 10" dance hit versus a lower listed position, the best model that was produced has an accuracy of 60 percent on the training set and an accuracy of 87.5 on the testing set. Their research also demonstrates the capability of predicting a song's success using data science techniques. However, they only used a dataset with 400 observations. The limitation of the data adds structural and estimation errors to the model. In our project, we have a total of 41106 observations which is much more reliable to train a predictive model. Therefore, we expect to have a better model that can lean the training set well and generalize enough to the unseen data.

3 Data Description

3.1 Spotify

Spotify is one of the most famous music Apps in the world, a program of last generation grown up exponentially over the last few years. This platform allows you to listen to music streaming to computers, smartphones and tablets, choosing from more than 30 million tracks, old and new, of the main international record companies, without having to purchase individual songs or albums legally.

3.2 General Information

The dataset that we plan to use consists of features for tracks fetched using Spotify's Web API. The tracks are labeled "1" or "0" ("Hit" or "Flop") depending on some criteria of the author. This dataset can be used to make a classification model that predicts whether a track would be a "Hit" or not. (Note: The author does not objectively considers a track inferior, bad or a failure if its labeled "Flop". "Flop" here merely implies that it is a track that probably could not be considered popular in the mainstream.)

This dataset contains 41106 rows and 19 feature (13 numerical variables and 6 categorical variables) that can be used to classify the target variable, "hit" rate. With each row representing one song, the data covers songs from 60s in 20th century all the way to 10s in 21th century. The target variable is a balanced binary categorical variable with the number of songs in both classes being similar. The target variable can be either "0" or "1". "1" implies that this song has featured in the weekly list (Issued by Billboards) of Hot-100 tracks in that decade at least once and is therefore a "hit". "0" Implies that the track is a "flop". All columns do not contain missing values.

3.3 Feature Details

Below are detailed information of some of the features:

- track: The Name of the track.
- artist: The Name of the Artist.
- danceability: 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. A value of 0.0 is least danceable and 1.0 is most danceable.
- energy: 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: The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C?/D?, 2 = D, and so on. (If no key was detected, the value is -1.)
- loudness: The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.
- mode: 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.
- tempo: 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: The duration of the track in milliseconds.

4 Visualization

From figure 1, we can see the distributions of each predictor with respect to the target variable. This plot gives us relationships between a single predictor and the target variable. For example, there is a discernable pattern for energy variable. The mean of the hit song's distribution is higher, and the distribution is narrower compared to flop songs. This makes sense to use since recent hit songs tend to be more energetic. Moreover, for the valence variable, hit songs have a left-skewed distribution whereas flop songs have a right-skewed distribution. High valence means more positiveness in the song which also seems to make sense in the real world. It's shown that danceability, energy, and valence are features that differs the most between the successful class and unsuccessful class.

In figure 2, we show the heatmap of the Pearson correlation of the numerical features. We can see

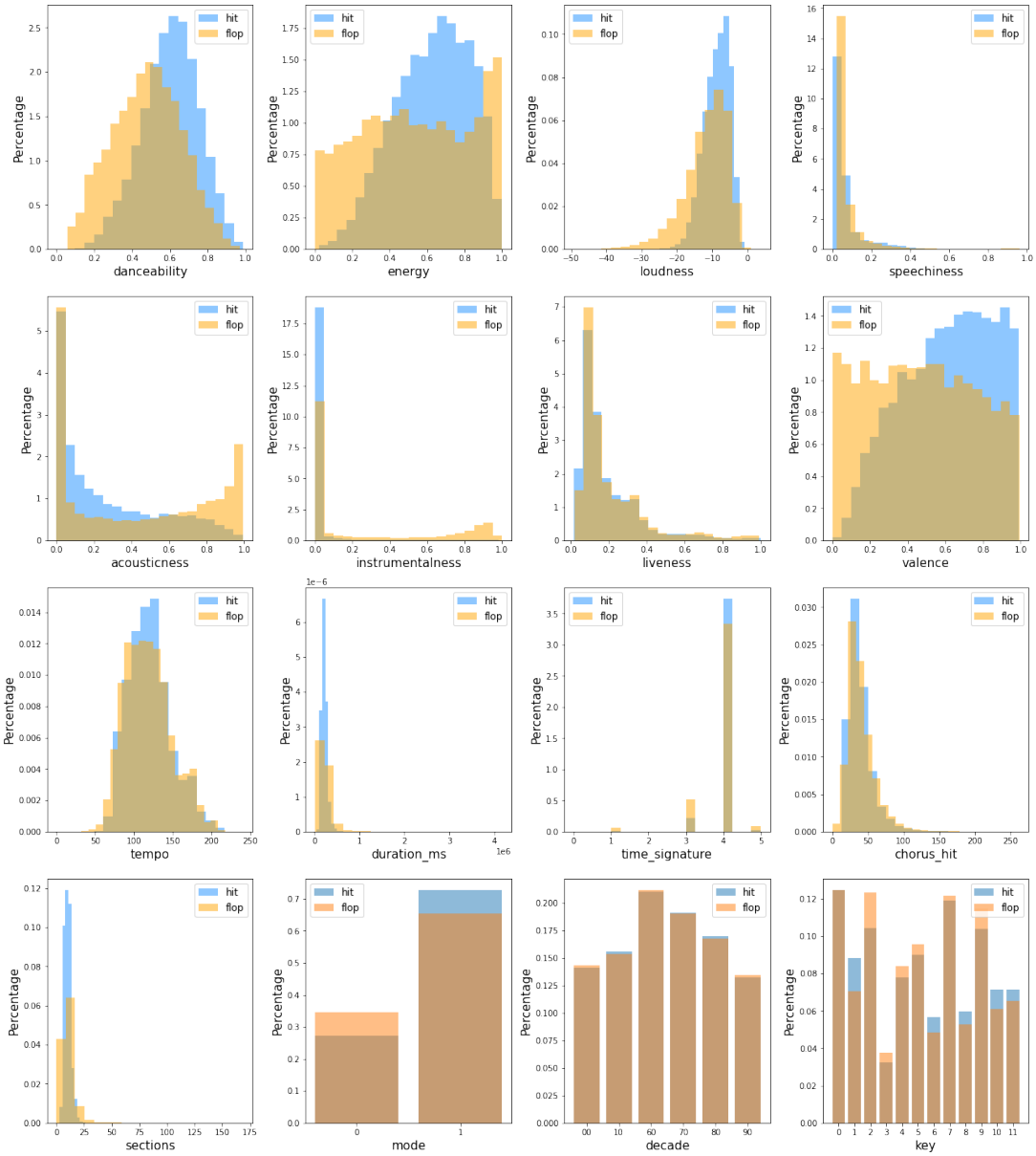

Figure 1: Distributions of the predictors with respect to the target variable

that some variables are highly correlated, such as sections and duration(ms), energy and loudness, and so on. Therefore, we will also consider the dimension reduction method. We performed the principal component analysis with the numerical data. In figure 3 shows the explained variance ratio versus the number of principal components used. Based on the results, it might be useful to consider the dimension reduction method since we are dealing with 41,106 data.

5 Problem Statement

Based on the observations above, we want to answer the following research questions:

- **RQ1** What features are the most correlated with song success rate?
- **RQ2** How to predict the song success rate given a song's meta information? How does our model compare with other baseline models?

6 Baseline Model

In our project, we implemented Logistics Regression, K-nearest Neighbor, and Decision Tree models (all the model parameters are using default values implemented in scikit-learn). We calculated the mean and the standard deviation of the training accuracy from the results of the 10-folds cross-

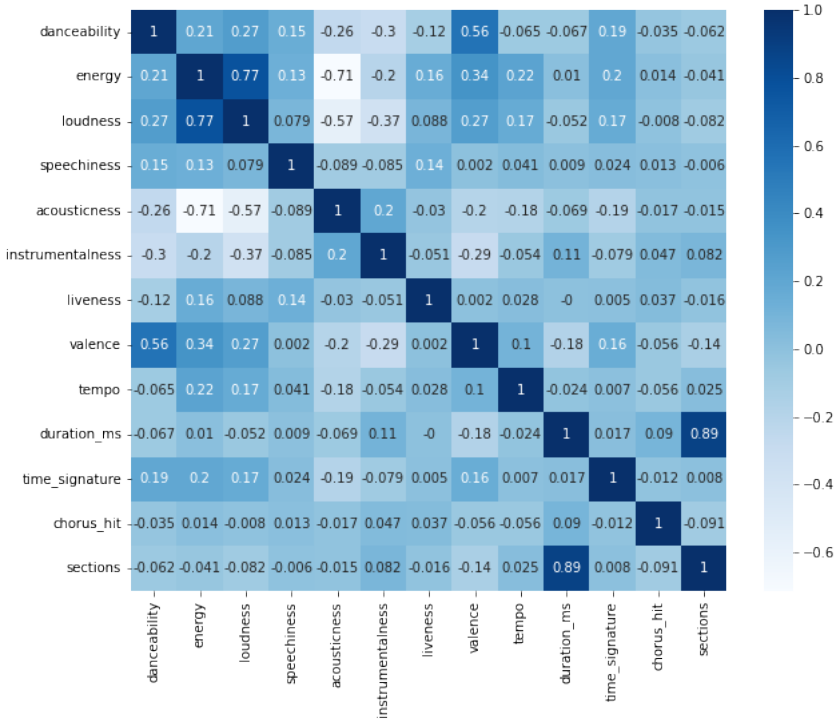

Figure 2: Heatmap of the correlation matrix

validation. We also calculated the model accuracy on the test set. The results are summarized in Table 1.

7 Methodology

7.1 Logistic Regression (LR)

We start with logistics regression which models the probability of an event taking place by having the log odds for the event be a linear combination of one or more independent variables. The reasoning behind this decision is that logistics regression is one of the simplest used for classification problems, has good interpretability, and can be trained very fast so that we can get the results quickly. Therefore, we start with the logistics regression model.

Although logistics regression has a number of benefits, it has drawbacks as well. Its linear assumption between the dependent and independent variables makes it only capable of creating a linear boundary. However, we do not expect to see perfect linear relationships in real-world applications. Therefore, we will also consider models that can model non-linear relationships in the following section such as K-nearest neighbors, trees, and neural networks.

7.2 K Nearest Neighbors (KNN)

We then tried the K Nearest Neighbors classifier, which makes predictions based on the k-closest data points. We chose this model because it is also a simple predictive model since it uses lazy learning, no assumptions, and few hyperparameters. Moreover, another benefit of using KNN is that we have a balance labeled dataset based on the results from EDA. If we have an unbalanced dataset, we will run the risk of KNN always giving a prediction of the majority label. Aside from these advantages, KNN is slow in prediction, and we need to be careful of the outliers and the curse of dimensionality.

7.3 Decision Tree (DT)

The decision tree classifier was then employed. We especially want to use it to handle the non-linear relationship in the data. Decision trees take less work to prepare the data during pre-processing than other methods do. The normalization and scaling of data are not necessary when using a decision tree. Technical teams and stakeholders can understand a decision tree model very quickly. While for the drawbacks, a slight change in the data might result in a huge change in the decision tree's structure, which can lead to instability. When compared to other algorithms, a decision tree's calculations

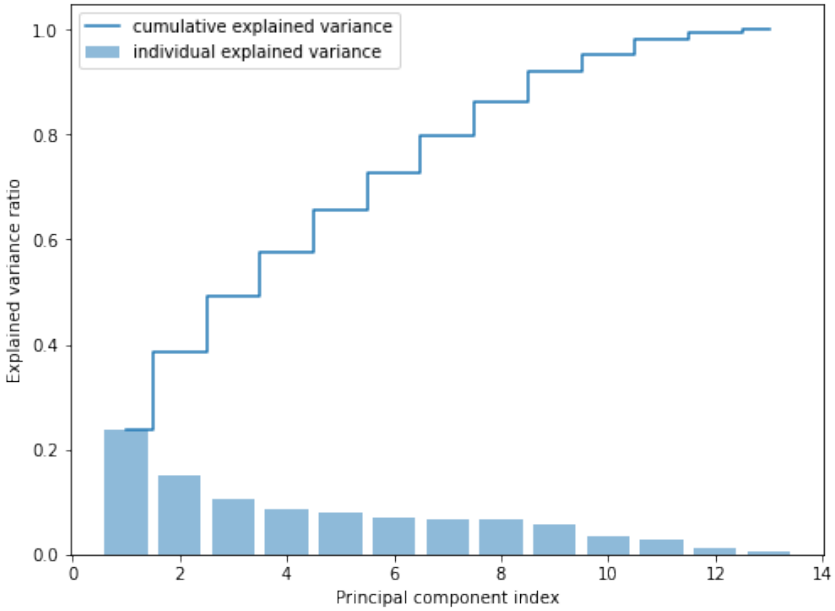

Figure 3: PCA for the features

Model	Mean Train Accuracy (10-folds cv)	Std Train Accuracy (10-folds cv)	Test Accuracy
LogReg	0.729	0.009	0.731
KNN	0.744	0.005	0.744
DT	0.714	0.01	0.714

Table 1: Baseline Model Performance

can occasionally become significantly more complex. The model training process for decision trees typically takes longer for our data. Because of its intricacy and lengthier training period, we can conclude that decision tree training is relatively expensive.

7.4 Random Forest (RF)

Based on the positive results from the decision tree classifier, we tried the random forest classifier, which is a forest of trees. While decision trees have good prediction power, it is subject to the issue of overfitting. Therefore, we used the random forest, which is an ensemble method that uses bootstrapped data to fit the individual decision tree. RF does not change the bias of the model a lot but works well in reducing the prediction variance. The main drawback of using RF in our project is that we have a large dataset that takes a long time to train RF and perform parameter tunings.

7.5 AdaBoost

To increase the efficiency in binary classification, we attempted to include the implementation of an AdaBoost classifier. Different from RF, AdaBoost adaptively weights the difficult-to-classify samples more heavily to boost the performance of weak learners. Then, it combines weak observations with strong learners to generate the final classification decision. Given its fairly good generalization, we chose this model for training our data. While it reduces both variance and bias and is easy to train, it is sensitive to outliers in noisy data.

7.6 Stacking

Besides training each individual model, we implemented Stacking because it allows the combination of heterogeneous models to reduce bias. Specifically, stacking improves model predictions by ensembling outputs made from multiple models and running them through a meta classifier. As models mentioned previously might make different assumptions on our data, we decided to use this method to examine performance across our complex and diverse choices of models. However, due to the stacking of different models, this approach is more computationally heavy than other models.

7.7 MultiLayer Perceptron (MLP)

In addition to models ensembled in Stacking, we also trained a MLP model to make predictions through neural networks. As our data is fairly large and there might be underlying nonlinear relationships, we decided to use MLP to provide predictions. However, we were aware that there is little interpretability in neural networks, and it is more expensive to train MLP.

8 Results

8.1 Logistic Regression (LR)

We used the LogisticRegression from scikit-learn and performed parameters tuning with the norm of the penalization as well as the inverse of regularization strength. For the 'l1' penalty, the coefficients of less contributive features are shrunk to zero and only the most significant features are kept. For the 'l2' penalty, all the features are included in the model, but variables with minor contributions have their coefficients close to zero. Lastly, the 'elastic net' penalty is a combination of the 'l1' and 'l2' penalties. We tried a wide range of penalization strengths from 0.01 to 100. The best set of parameters using 10-folds cross-validation used a penalty strength of the inverse of 0.01 and an 'l2' penalty. This makes sense to us that we want a relatively large penalization strength to prevent overfitting issues in linear regression. We also want to include all the predictors in the model since we only have 15 features. Moreover, we found that the 'valence' variable has a positive coefficient which means that an increase in the musical positiveness conveyed by a track will lead to an increase in the log odds of the hit in prediction, which matches our EDA result.

8.2 K Nearest Neighbors (KNN)

We used the KNeighborsClassifier from scikit-learn and performed parameter tuning with the number of neighbors, weight function for prediction, and the distance metrics. The best 10-fold cross-validation result used 16 nearest neighbors, distance as weights, and the Manhattan distance. This makes sense to us that we want to use a relatively large number of K to prevent overfitting issues. We also expect the Manhattan distance to work better than the Euclidean distance since we are in a 15-dimensional space.

8.3 Decision Tree (DT)

We used the DecisionTreeClassifier from scikit-learn and first fit the model using all the default parameters. We found the max depth of the tree to be 34 which provides us with insight into how to do parameters tuning next. The best 10-fold cross-validation result used a max depth of 10, and a min sample split of 1 percent of the data, which makes sense to us that we want to prune the decision tree to prevent overfitting issues. However, a single decision tree did not improve the learning and prediction power a lot compared to logistics regression and KNN. Therefore, we will consider advanced tree models later to remedy this issue.

8.4 Random Forest (RF)

We used the RandomForestClassifier from scikit-learn and performed parameter tuning with the number of estimators, max depth for a single tree, and the minimum sample split. The best 10-fold cross-validation result used 200 estimators, a max depth of 11, and a min sample split of 0.01 percent of the data, which makes sense to us that the selected parameters are similar in DT and we also want to prune the decision tree to prevent overfitting issues. As we expected, the RF model has a three percent accuracy improvement in the testing set since bootstrapped data and 200 estimators can reduce the variance.

8.5 AdaBoost

We first trained AdaBoost with default parameters from scikit-learn. After tuning of hyperparameters including max_depth and learning_rate, we decided the best model to have a base estimator of DT with a max depth of 3, a learning rate of 0.3, and 150 estimators. This model achieved an accuracy of 0.80, the same as what we achieved in RF.

8.6 Stacking

We performed hyperparameter tuning for Stacking to find the best parameter combinations. The base estimators include LR, KNN, DT, RF, and AdaBoost that we mentioned previously. The best performing combination yields an accuracy of 0.80. This does not outperform any of our previous models, probably due to the high number of overlapping in making correct predictions from the ensembled models.

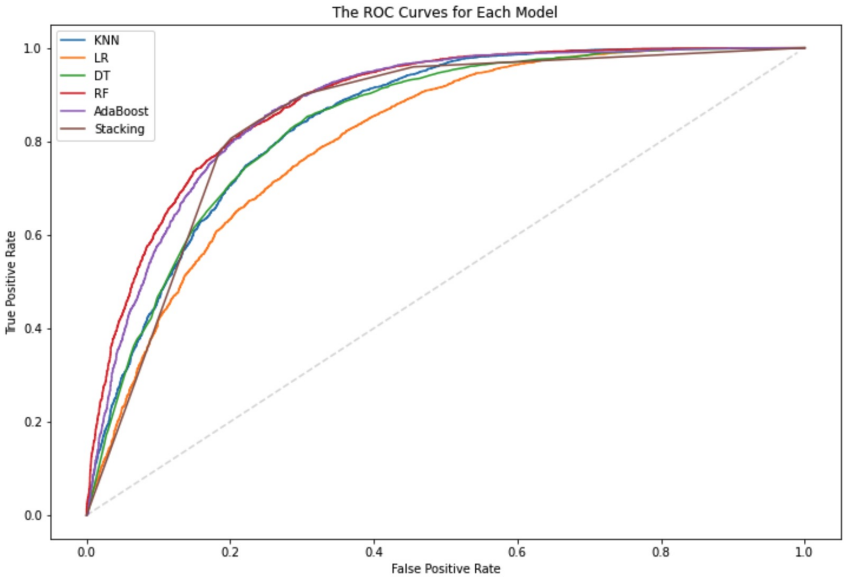

Figure 4: ROC of different models

Model	Training Time	Precision	Recall	F1-Score	Accuracy
LR	0.23s	0.74	0.73	0.73	0.73
KNN	0.01s	0.79	0.75	0.75	0.76
DT	0.22s	0.78	0.77	0.77	0.77
RF	25.1s	0.80	0.80	0.80	0.80
AdaBoost	14.26s	0.81	0.80	0.80	0.80
Stacking	55.43s	0.81	0.80	0.80	0.80
MLP	365s	0.80	0.80	0.80	0.80

Table 2: Model Comparison across Evaluation Metrics

8.7 MultiLayer Perceptron (MLP)

The best parameters for the MLP model had the tanh activation function, a hidden layer size of (150, 80, 40), a max iteration of 100, and a solver of adam. This yields an accuracy of 0.80 for 10-fold cross validation and a final loss around 0.43, which is about 0.7 lower than what the baseline model had. The precision and recall were the same as accuracy, indicating balanced true positive and false positive rates.

8.8 Model comparison

To compare model performance from the tree family, including DT, RF, and AdaBoost, we focused on the comparison of their relative variable importance as shown in Figure 5. We see that while *instrumentalness* is the most improtant feature in decision tree and random forest, it is not of much importance in AdaBoost. The feature *acousticeness*, in contrast, is important across all models.

In addition to comparing feature importance, we constructed ROC curves (Figure 4) to visualize trade-offs between true positive and false positive rates for Stacking and all models used in Stacking. We see that RF and AdaBoost presented similar performance, and Stacking achieved almost the same performance with RF and AdaBoost. All these three models, including RF, AdaBoost, and Stacking, performed better than LR, KNN, and DT according to the graph.

To provide comprehensive results for different potential use cases of our model, we compared our experimental models across multiple evaluation metrics, including training time with best parameters, precision, recall, f1-score, and accuracy, as shown in Table 2. For instance, precision score may be used when music investors are curious about the true rate of correctly predicting successful songs. The evaluation on training time for each model with their best parameters help us understand advantages and disadvantages for each experiment.

From Table 2, we see that The best performing models are RF, AdaBoost, Stacking and MLP, as all of their metrics are on or above 0.80. This

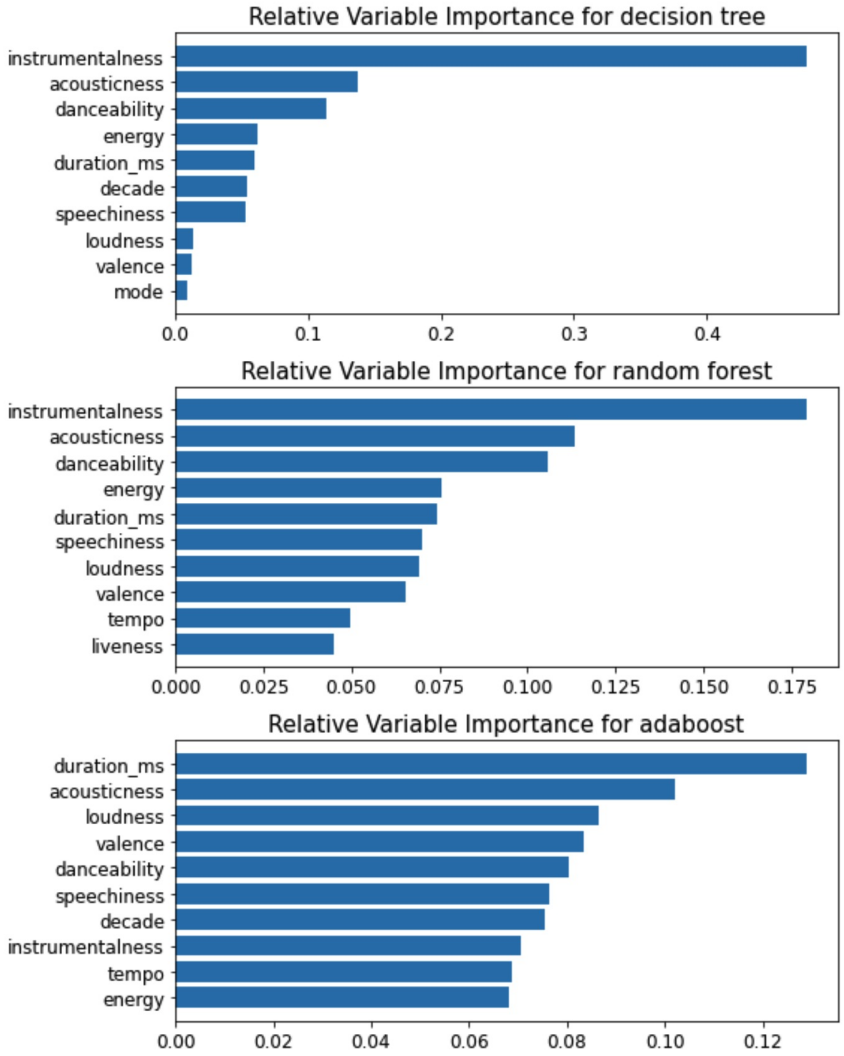

Figure 5: Important features picked by different models

greatly exceeds accuracies achieved in our baseline models. Among these, MLP is the most expensive to train. The recall scores are the same for RF and AdaBoost, indicating the same ability of making correct song hit predictions. AdaBoost and RF have the same performance across all metrics, but RF is slightly more expensive to train.

For the rest of models, results also make sense. For example, since KNN is free from linear assumption, we found KNN performs better than logistics regression in both training and testing accuracy as we expected. A single decision tree also does not seem to help in capturing the complex underlying relationships in our data.

In conclusion, AdaBoost is the best model because it does not only yield the highest performance metrics but also obtains the shortest training time.

9 Conclusion

We conducted exploratory data analysis on The Spotify Hit Predictor Dataset, identifies important features for song success prediction, and observed that danceability, energy, and valence are features that differs the most between the successful class and unsuccessful class. We then experimented various machine learning models with regularization methods for song success prediction task. Among those models, Random Forest and AdaBoost outperform the rest by both achieving 0.8 F-1 score and 0.8 accuracy score. The recall scores are the same for both Random Forest and Adaboost, meaning that among all the successful songs, the two models make the same amount of correct predictions. On the other hand, AdaBoost achieves slightly better precision score on song success prediction, implying that Adaboost model can be more useful for music investors who cares more about

how many songs predicted as success are actually success.

10 Future work

In the future, we want to incorporate more features into the dataset from more sources, not just limited to Spotify, and examine the performance of deeper neural networks on the song success prediction task. For example, analyzing textual lyrics may help in understanding how genres and song themes are associated with song hits. We would also like to use kernel-based framework such as Support Vector Machine to improve the flexibility in classification. What's more, since the dataset is not the most up-to-date, we want to examine the performance of the best model on currently trending songs. Last but not least, we may extend our project to provide a user-friendly tool that provides insights of song success for music investors.

11 Acknowledgments

We are grateful to our professor Dr. Pavlos Protopapas and Dr. Natesh Pillai for their fantastic lectures that covers the most important area in data science. Thanks should also go to all the TF's of AC 209A who provide enormous support and help along the way.

References

Ruth Dhanaraj and Beth Logan. 2005. Automatic prediction of hit songs. In *ISMIR*, volume 11, pages 488–491. Citeseer.

Dorien Herremans, David Martens, and Kenneth Sörensen. 2014. Dance hit song prediction. *Journal of New Music Research*, 43(3):291–302.

Junghyuk Lee and Jong-Seok Lee. 2018. Music popularity: Metrics, characteristics, and audio-based prediction. *IEEE Transactions on Multimedia*, 20(11):3173–3182.

Yizhao Ni, Raul Santos-Rodriguez, Matt Mcvicar, and Tijl De Bie. 2011. Hit song science once again a science. In *4th International Workshop on Machine Learning and Music*.

Eva Zangerle, Michael Vötter, Ramona Huber, and Yi-Hsuan Yang. 2019. Hit song prediction: Leveraging low-and high-level audio features. In *ISMIR*, pages 319–326.