Using Spotify to Predict Billboard Top 100 Songs

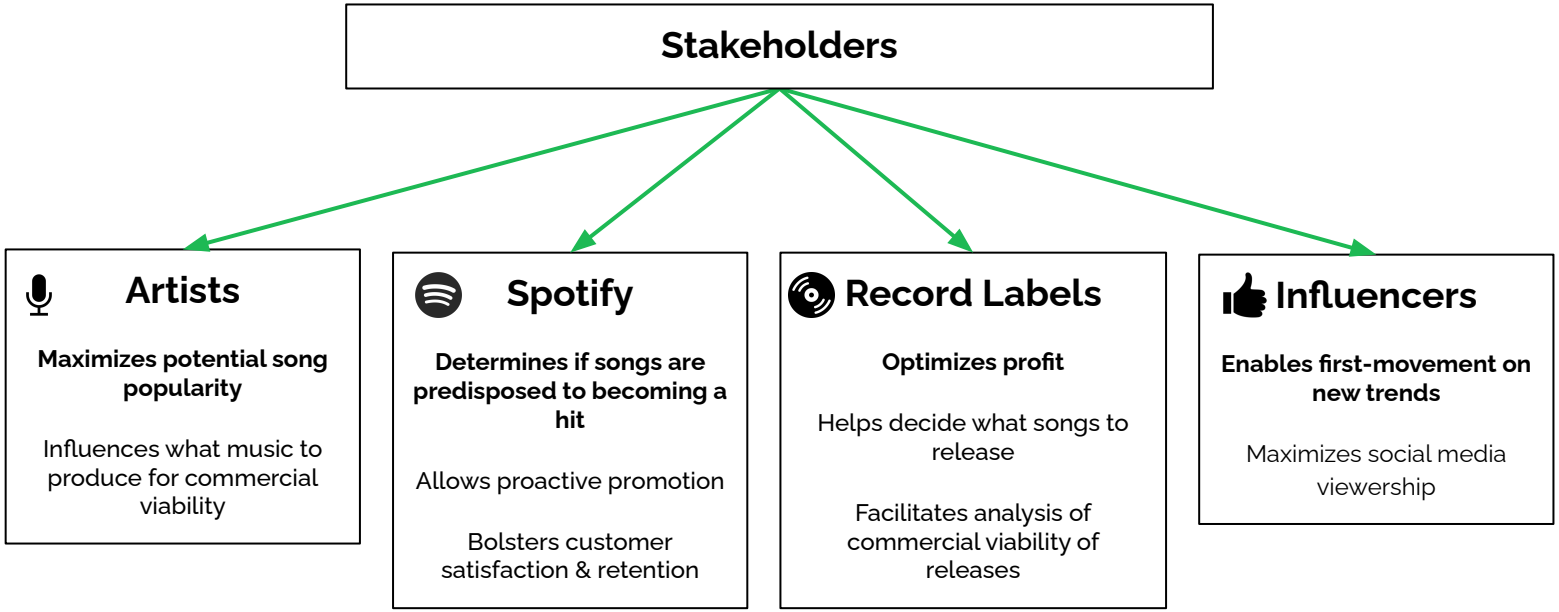
BA476
Professor Gerdus Benade
Team 2: Tristan Tew, Claire Choi, Rebecca Chang, Brett Rado

Executive Summary

- The Problem & Stakeholders
- Data Sourcing, Transformations, & Cleaning
- Descriptive Analytics & Observations
- Predictive Methods & Model Accuracies
- Obstacles & Response
- Conclusions

Predicting Commercial Viability for a Song Creates Value for Multiple Parties

The Problem: ***Predicting whether or not a new song will end up on the Billboard Top 100 list based on its Spotify information***



Kaggle and Python API Allow us to Add Over 300 Features to the Model

The Dataset: The Spotify Hit Predictor Dataset for 2010's (Kaggle)

Raw Dataset:
- Rows: 6398
- Predictors: 15
- Types: Numerical, Boolean, Categorical

Cleaning Process:
- Added features to songs from Spotify API
- Generated 312 from Binary Genre Variables
- Added polynomial features and converted categorical variables

Cleaned Dataset:
- Rows: 6344
- Predictors: 374
- Types: Numerical, Boolean,
- Train/Test: 80/20

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	...	australiancountry
0	0.741	0.626	1	1.762331	0	11.286682	0.271442	0.000000	2.491327	0.7060	...	0
1	0.447	0.247	5	2.751174	0	28.901734	0.955006	0.949852	2.358098	0.2500	...	0

Handling of Null Values and Outliers with Log and Power Transformations to Reduce Skewness

Null Values and Removal of Rows:

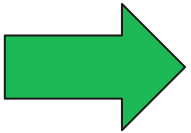
- 54 rows were removed from the dataset because their genres column was blank
- There were **no null values** the final dataset

Outlier Wrangling:

- |Skewness| > 1 as the rule for outliers for 16 numerical features
- Log, power, and reciprocal transformations were applied to mitigate skewness

Skewness Before

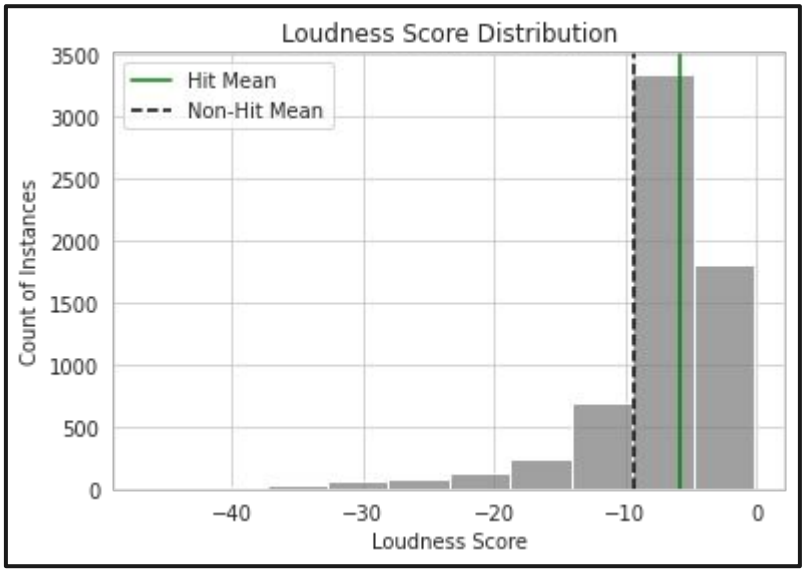
	Skew
loudness	-2.666130
energy	-0.860043
danceability	-0.382247
valence	0.178443
position_on_album_ratio	0.270647
tempo	0.323195
acousticness	1.450153
instrumentalness	1.626184
chorus_hit_percent	1.778256
chorus_hit	2.042746
liveness	2.238090
speechiness	2.370151
track_number	3.018350
duration_seconds	3.381844
sections	3.435853
album_tracks	3.785306



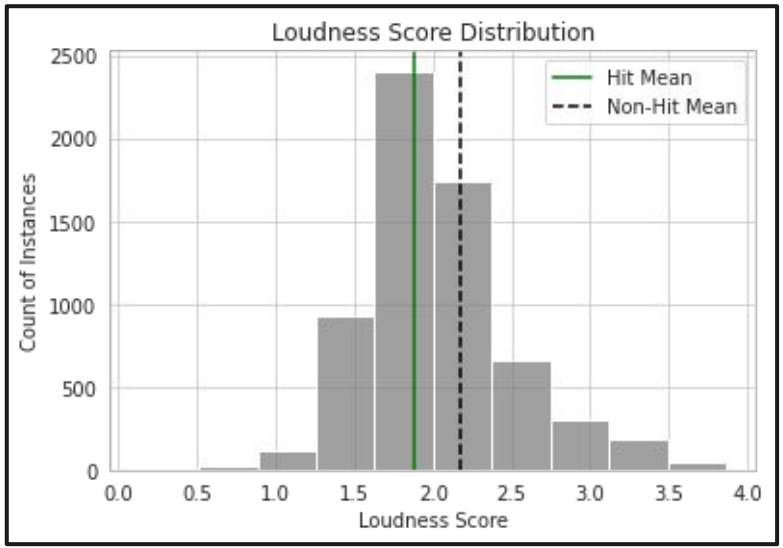
Skewness After

	Skew
album_tracks	-0.947025
energy	-0.860043
liveness	-0.527066
chorus_hit_percent	-0.414871
danceability	-0.382247
sections	0.117247
duration_seconds	0.131235
speechiness	0.174581
valence	0.178443
position_on_album_ratio	0.270647
tempo	0.323195
acousticness	0.337337
track_number	0.373839
loudness	0.779015
chorus_hit	1.006078
instrumentalness	1.061716

Transformations Example: Log




```
f(x) log(1 + -x)
```



Addition of Polynomial Features, Transformation of Genre Column, and Creating Dummy Variables Rounds out Data Manipulation

Genre Columns:

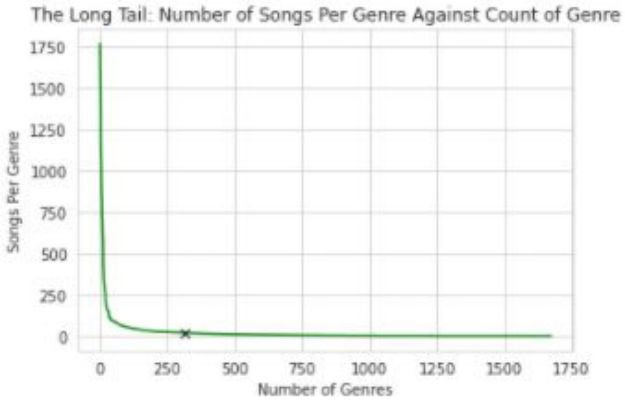
- Converted column of string representation of lists into **312 binary predictors**
- Created a dictionary of genre frequencies throughout the dataset and took the mean (21.7)
- Created binary predictors for all genres that occurred at least 21 times

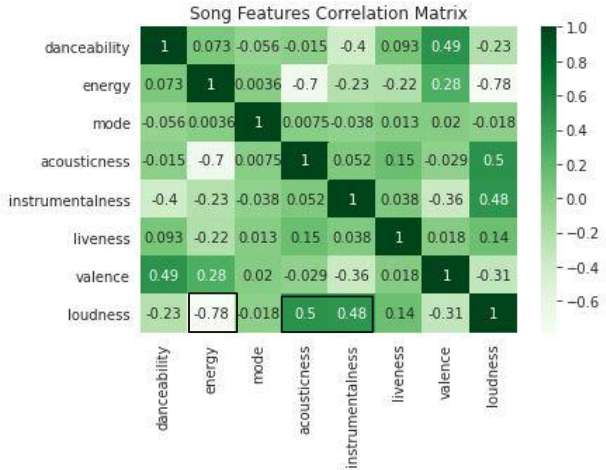
Polynomial Features:

- Began with the intuition and song feature correlations
- Chose song **loudness**, **energy**, **acousticness**, and **instrumentalness** to take to **second degree**

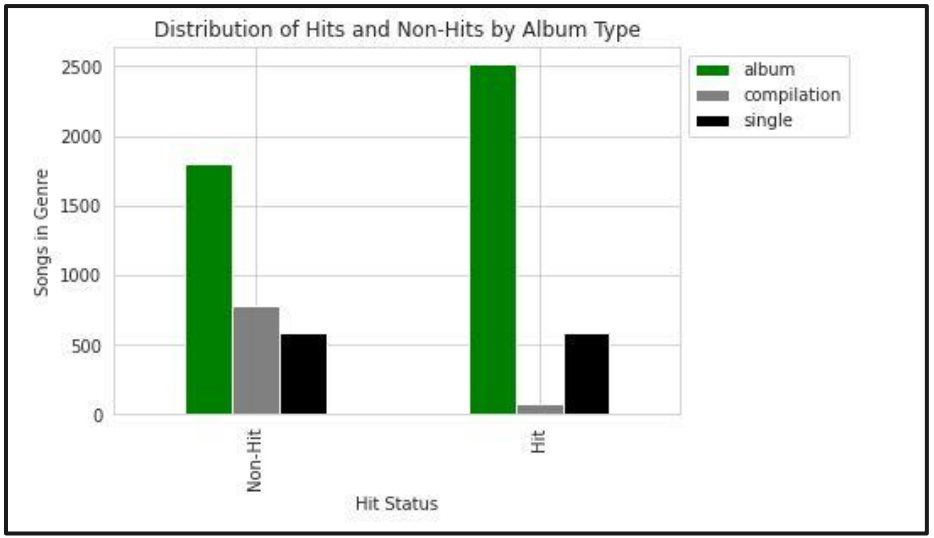
Categorical Variables:

- Song **key**, **time, release month, album type** converted via pd.get_dummies()

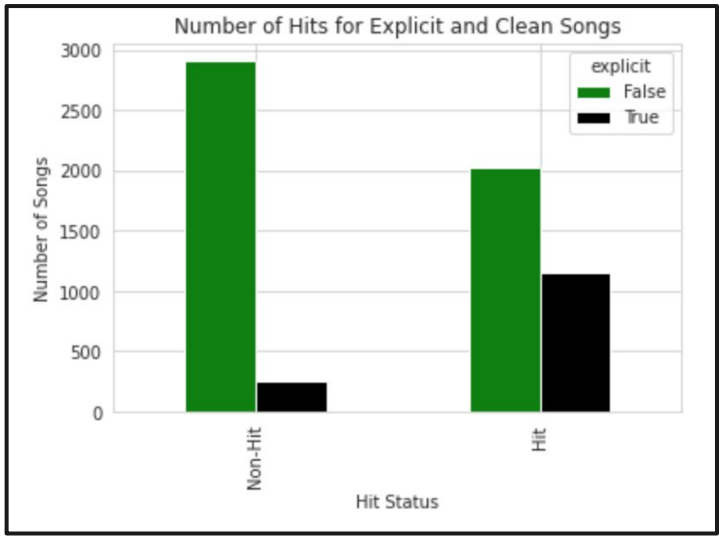




Album Status and Vulgarity Differentiate Instances Between Classes

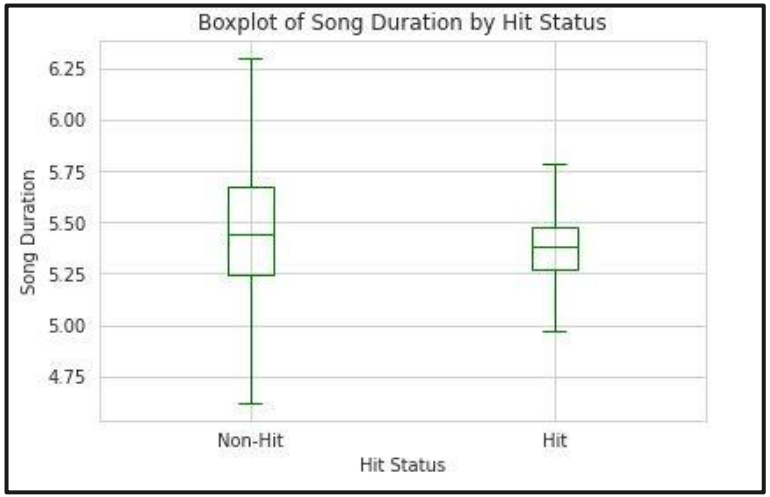


- Few hits come from compilation albums
- Hits and non-hits come from singles roughly equally
- Hits appear more on albums than others

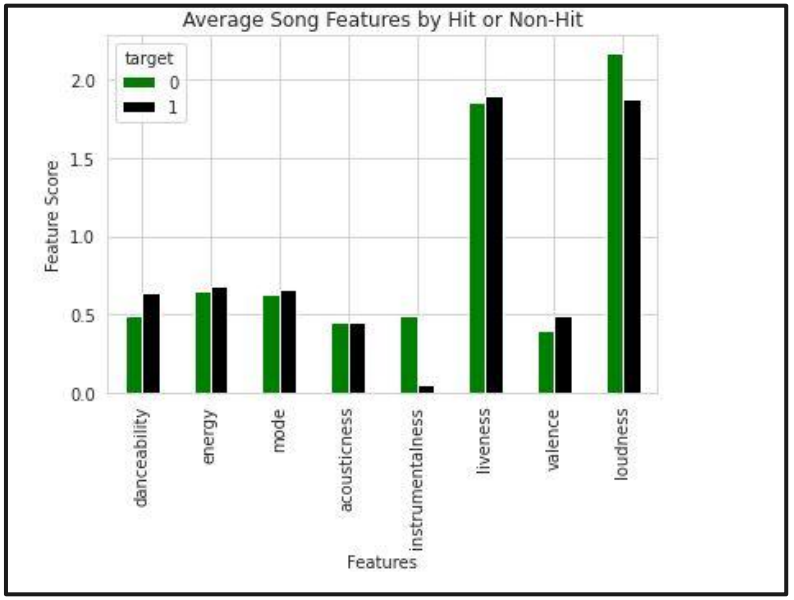


- Explicit songs have a high likelihood of being hits
- Clean songs may be hits, but the odds of being a hit go down

Song Features and Subsequent Distributions Further Differentiate Classes

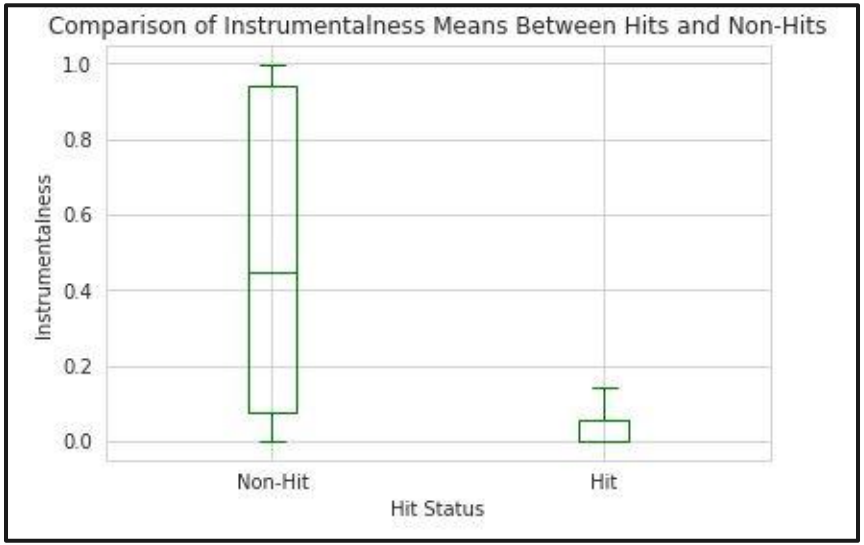


- Despite comparable means, non-hits have a larger standard deviation for durations



- Non-hits have higher loudness scores and instrumentalness scores on average
- Hits have a slightly larger valence and danceability score

Descriptive Analytics Reveal Trends Related to Later Feature Importances and New Features Drive Best Non-Stacked Model



- Means and standard deviations of instrumentalness scores differ.
- This reveals the possibility of instrumentalness as a important feature when building models

Feature Importances for Best Non-Stacked Model	
instrumentalness²	0.135061
instrumentalness	0.126435
loudness instrumentalness	0.110072
pop	0.065279
country	0.048886
rap	0.046902
alb_type_compilation	0.045935

- Variations of the instrumentalness score are the three most important variables in our best model that is not stacked
- 6 of the top 7 variables for this model were the result of feature engineering

Pre-Tuning Models Show Significant Performance above the Baseline

We employed logistic regression and the ridge classifier to see if regularization would solve our problem

We employed ensemble methods to maximize performance

	Baseline	LogReg	Ridge Classifier	Decision Tree	KNN	Random Forest	Bagging	Boosting	**Stacking**
No Tuning	**51.46%**	93.30%	92.36%	86.13%	89.60%	91.10%	93.22%	92.67%	**94.33%**

Our baseline was determined using the stratified dummy classifier

We employed non-parametric solvers as a tool to see if it better fit our data and establish a baseline for the tree-based ensembles

GridSearchCV Beats out RandomizedSearchCV for Parameter Tuning and the Stacking Ensemble Performs Best

	Baseline	LogReg	Ridge Classifier	Decision Tree	KNN	Random Forest	Bagging	Boosting	**Stacking**
Random SearchCV	+0%	**+0.08%**	-0.08%	+2.84%	-1.26%	+2.75%	+0.47%	+0.10%	-0.32%
Grid SearchCV	+0%	+0%	-0.08%	+2.84%	**+0.16%**	**+2.90%**	**+0.87%**	+0.10%	**+0.08%**
Best Model Accuracy	51.46%	93.38%	92.36%	88.97%	89.76%	94.01%	94.09%	93.77%	**94.41%**

For parameter tuning, we ran both GridSearch and RandomizedSearch to improve performance

- GridSearch took a long time, but returned the best results after running a RandomizedSearch on a small parameter range
- RandomizedSearch was faster, but since it only looks at a few random combinations of parameters, it did not produce the best result on its own

Overfitting Likely Explains Why Stacking and Bagging are the Best Ensembles

Among the ensembles, the best performing models overfit the least

- Bagging hyperparameters led to lower variance than the random forest
- Stacking with both similar and regularized models introduced different predictions to reduce overfitting
- High number of features in our final random forest or high learning rate may have led to the overfitting observed in the model

	Model Name	Test Accuracy	Train Accuracy	Difference
8	STCK	0.944050	0.998227	-0.054176
7	BAG	0.940898	0.999606	-0.058708
5	RF	0.940110	0.999606	-0.059496
6	BST	0.937746	0.998030	-0.060283
1	LogReg	0.933018	0.946798	-0.013780
2	RC	0.922774	0.940296	-0.017522
4	KNN	0.897557	0.999606	-0.102049
3	DT	0.889677	0.999606	-0.109929
0	BL	0.514578	0.508177	0.006401

The regularized models had the least overfitting, but performed worse

After Struggling with Genres Initially, they Solved our Biggest Roadblock

At first, we struggled to parse out genres from the song instances to create dummy variables.

- The genres were placed in the CSV as strings, not lists, requiring extensive processing.
- We created a dictionary of all genres and then added features to our model based on whether or not the genre name was found in the genre column.

After discussing with Professor Benade, we realized many of our predictors were unreliable.

- For example, artist popularity reflects the current popularity of an artist, not the popularity of the artist when a song debuted.
- After removing these predictors, we lost ~10% of our predictive accuracy (95% → 84%).

To combat unrepresentative predictors, we added 302 more genre predictors. This increased predictive accuracy by 10% (84% → 94%) to negate the loss of excluding those predictors

Training the Spotify Models Has Challenged Us to Consider Predictors and Alternative Tuning Methods

Choosing predictors is not just about accuracy, but also building structurally sound experiments

- We learned that variables must be a suitable representation of the data and that time; thinking about predictors without context can create misleading models
- Using information from only Spotify only tells us part of the story, finding data from other platforms would be a next logical step to address limitations that come with suitable data

In future projects, we will consider the context of the predictors for our stakeholders' sake

Depending upon time constraints and the project, different tuning methods may be more appropriate

- Using RandomizedSearchCV to determine hyperparameter estimates to pass into GridSearches allowed us to achieve optimal accuracy
- RandomSearchCV can be a good tool to give relative hyperparameter ranges for the more granular GridSearchCV to explore

In future projects, we will use the timeline as a guiding principle for which method to select, but if time permits, we will likely use both in tandem to produce the optimal model