

# Predicting Hit Songs

Henry Chen, Cecilia Wu, Yiyuan Xu, Timothy Yang, Zheng Zhou

## ABSTRACT

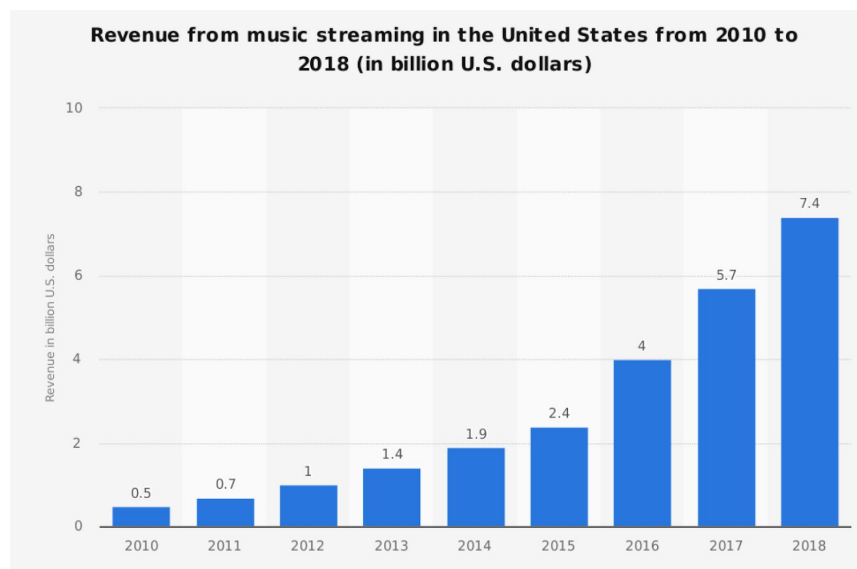
Exploring the possibility of predicting hit songs is both interesting from a scientific point of view and something that could be beneficial to the music industry. In this research we raise the question if it is possible to classify a music track as a hit or a non-hit based on its audio and artist features. By training classifiers on the different features, we hope to find patterns in the data that will hopefully tell us which features are appealing to people and will result in a song becoming popular, or “hit”. We investigated which machine learning algorithms could be suited for a task like this. Four different models were built using various algorithms, including K Nearest Neighbors, Support Vector Machines, Naive Bayes, as well as Logistic Regression. The obtained results do not indicate that it is possible to predict hit songs on our particular dataset. (Testing on the full imbalanced data gives us worse results than the resampled data did). The fact that we predict most of the songs as hit songs results in a high recall, and precision becomes very low because of it. Also, we do not get a conclusion of what the most relevant features are because they vary from model to model.)

**Keywords:** Hit song, Binary Classification, Musical features

# INTRODUCTION

## Background

With the growing presence of music streaming services such as Apple Music and Spotify, US music streaming revenue has reached 7.4 billion in 2018. Apple Music alone has achieved 60 million monthly subscribers in June 2019, while Spotify holds the lead with 113 million monthly subscribers in the third quarter of 2019 (Newcomb).



The number of users are growing, but the payout rates are not. Apple music for example, pays about \$0.00735 each time a song is streamed, and Spotify pays around \$0.00437 per play. A famous artist is expected to be paid more, but the variance is small (Sanchez). According to a study done by UC Davis, “the annual earnings for a full-time minimum-wage worker is \$15,080 at the current federal minimum wage of \$7.25 (UC Davis)”. To earn the minimum wage, an artist must have their song streamed 2.05 million times on Apple Music, or 3.45 million times on Spotify. For an average artist, it is always a gamble not knowing whether or not your next track

will make you enough to put food on the table. It is in our best interest in helping artists benefit from the booming streaming industry.

## Motivation

Our goal is to train a machine learning classifier to determine whether or not a song will be a “hit song”, defined as the song having made the Hot 100 list on Billboard, from the musical and artist features of a song. By doing this, we hope that artists will be able to determine if a song has the potential to become a hit song early on in the process. This will alleviate the risk of wasting time, money, and effort into producing a song that the audience dislikes. By identifying the features that are common among hit songs, we can guide artists to pay more attention on those features while brainstorming a song.

## Terminology

**Overfit:** a modeling error that indicates the model is training data too well. It often happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.

**Regularization:** “a weapon to combat overfitting and underfitting. It constrains the learning algorithm to improve out-of-sample error, especially when noise is present.” (Abu-Mostafa )

**Kernel:** Kernel methods are a class of algorithms for pattern analysis, whose best known member is the support vector machine (SVM). The general task of pattern analysis is to find and study general types of relations (for example clusters, rankings, principal components,

correlations, classifications) in datasets.

**K Nearest Neighbors:** a type of instance-based learning or non-generalizing learning. It does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the k nearest neighbors of each query point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

**Support vector machines (SVMs):** a set of supervised learning methods used for classification, regression and outliers detection. They are effective in high dimensional spaces. They use a subset of training points in the decision function (called support vectors), so they are also memory efficient. However, if the number of features is much greater than the number of samples, over-fitting must be avoided by choosing Kernel functions and regularization constant.

**Naive Bayes:** a supervised learning algorithm based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.

**Logistic Regression:** a linear model for classification rather than regression. In this model, the probabilities which are numbers between 0 and 1, and describe the possible outcomes of a single trial are modeled using a logistic function.

**Ein:** The error rate obtained by testing on the training data. In our case, this is obtained by testing the classifier on the resampled data.

**Eout:** The error rate obtained by testing on the testing data, or unseen data. This gives a

good estimate towards how the classifier will perform in the real world. We calculated this by testing our classifier on the testing data split, as well as the unbalanced dataset.

## Prior Work

Before diving into related works, we want to provide some background on The Echo Nest as many related works used their platform for obtaining data about audio features during their research. The Echo Nest is a leading music intelligence company that has over a trillion data points on over 38 million songs in its database. In 2014, Spotify announced that they had acquired The Echo Nest. Some of the audio features derived by The Echo Nest for audio tracks are still available through the Spotify API but we noticed during our project that a great number of information that was earlier accessible unfortunately is no longer available for us to use.

Previous attempts have been made to address the question if it is possible to predict if a song will be popular. However, researcher had diverging outcomes on whether music features can predict the popularity of a song.

Pachet and Roy from Drexel University conducted a research based on a dataset of 32000 song entries and 632 features to see if the popularity of songs can be predicted from acoustic or human features, and published their results on the International Conference on Music Information Retrieval. They were not able to develop a good classification model and concluded that the popularity of a song cannot be learnt by using machine learning (Pachet and Roy 2008).

Another group from Standard investigated if they could predict the popularity of a song based on its audio features and Youtube view counts. The features for audio tracks were obtained

from The Echo Nest. For this task, a number of Support Vector Machines were built. The Support Vector Machines, regardless of feature choice and parameters, never achieved more than 53% accuracy. They draw the conclusion that audio features alone do not seem to be good predictors of what makes a song popular. They suggest that popularity is likely driven by social forces (Borg and Hokkanen 2011).

Another project done by Keven Wang, also from Standard University, concluded otherwise. He extracted musical features from MIDI files and grouped features into categories to test on 1752 samples of 50%/50% balanced data. Logistic Regression was chosen as the best classifier, and an ensemble method peaked precision 0.882 at probability cutoff 0.998 (Wang 2014)

## **METHODS**

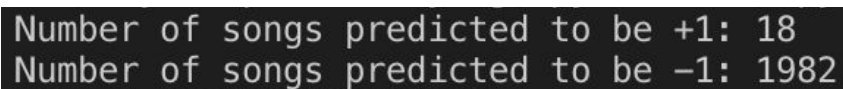
### Key Challenges

The dataset from the Million Song Dataset (MSD) source consists of 280GB of contemporary songs. As students working on personal PC's, we found that the size of the dataset and projected running time of classifiers to be impossible to work with. After a thorough investigation of other backup datasets, we concluded that the MSD provides a more reliable cleaned dataset with most music features, so we narrowed the scope and decided to only work with a subset of the enormous data.

After downloading a subset which contains 10,000 contemporary songs, we found out that only about 12% of those songs made to the Billboard Hot 100. This was troubling because

such dataset is imbalanced, meaning there is a large amount of observation for one class (majority class) and much fewer observations for the other class (minority class). An imbalanced data for supervised machine learning models is subject to a high risk of frequency bias, where the models will emphasis on learning observations that occur more common. In this case, classification models have a tendency to learn on those songs that don't not considered to be hit songs, but we want to understand the characteristics of hit songs. For example, upon testing on a 20% sample of imbalanced data, we got the following results in image 1.

*Image 1*



```
Number of songs predicted to be +1: 18
Number of songs predicted to be -1: 1982
```

As we can see, 99.1% of the songs were predicted to be non-hit songs. We proposed two solutions to the problem:

1. rebalance the data to have “hit songs” as the majority class by using a 66%/33% split from the original 12%/88% split
2. randomly resample the dataset to include fresh entries
3. used k-fold cross-validation classification
4. use confusion matrix to distinguish true positives

To avoid data snooping, we also included a test and training split of the dataset in additional to our 10-fold validation.

## Dataset

We used data from the Million Song Dataset (MSD) - a dataset with one million song entries available from the collaboration between LabROSA and the Echo Nest. The total size of one million song files is approximately 280 GB (gigabytes). Due to disk space restraints and also for the consideration of a reasonable training time of the classifiers, a subset of 10,000 songs is used for this work. The subset is provided by the MSD team, containing randomly selected data for training purposes.

Another dataset from data.world is also used for training. The lists of Billboard Hot 100 data were scraped from the website to label the 10,000 songs. By matching song names from the MSD and the Billboard Hot 100 lists, a merge set labeled each attribute: +1 if the song has made it to the Hot 100's and thus is a hit song, or -1 if it has not. Even though we resampled the dataset to have a total size of 1836 songs (1214 +1's, and 612 -1's) to prevent frequency bias, the models are tested on the full 10,000 songs to have a better approximation of what the performance will be like for the entire population.

### Feature Selection

From the song files, 15 features were extracted as shown in figure 1. All 15 predictors are quantitative continuous or ordinal variables.

Feature Name	Description
Artist Hotness	How famous is the artist currently
Artist Familiarity	How well known the artist is
Danceability	How suitable a song is for dancing
Duration	Duration of the song in seconds
End of Fade In	Time of the end of the fade in, at the



	beginning of the song
Energy	How “energetic” a song is
Key	An estimation of the key of a song
Key Confidence	The confidence of the key estimation
Loudness	General loudness of the song
Mode	An estimation of the mode of a song
Start of Fade Out	The start of the fade out at the end of a song
Tempo	Tempo is beats per minute
Time Signature	An estimation of the time signature of the song, i.e. number of beats per bar
Time Signature Confidence	The confidence of the time signature estimation
Year	The year the song was released

Fig 1. Predictors of the model

We compared the feature importance using the simple model Naive Bayes and here is the comparison of the features’ F1 ( top 12 are included )

Feature	F1	Feature	F1	Feature	F1
Year	1.0	Time Signature Confidence	0.8071	Energy	0.7954
Time Signature	0.8167	Loudness	0.8033	Start of Fade Out	0.7934
Mode	0.8111	End of Fade In	0.7993	Danceability	0.7926
Tempo	0.8091	Key Confidence	0.7974	Artist Familiarity	0.7759

Fig 2. Comparison of features’ F1

## Implementation

To work on codes and written works simultaneously, we made use of online tools such as Google Colab and Google Doc to comment on each other's work and increase efficiency.

We coded everything in Python 3.0, because the package scikit-learn was at our disposal for a variety of methods that are useful for machine learning, such as feature selection, training and testing data split, and classifiers. For instance, **sklearn.feature\_selection** module was used for feature selection.

Although all 15 features are considered for classification models, not all of them are significant in predicting hit songs. The **SelectKBest** function and **f\_classif** from scikit-learn were used to help select the best features with highest ANOVA F-values to reduce overfit and keep our models at a reasonable complexity. The **get\_support** function will show which features are chosen. In addition, other packages such as matplotlib.pyplot are also used to create visuals, so to demonstrate the performance of the different classifiers.

## Model Selection

We decided on using 4 classifiers:

1. Naive Bayes
2. K-Nearest Neighbors
3. Support Vector Machines
4. Logistic Regression

Naive Bayes is the simplest classifier chosen to serve as a baseline comparison.

K-Nearest Neighbors is based on feature similarity, and in this case where the features of hit songs should be similar, it is a good first choice for classification. Support Vector Machines are very effective in cases where there are many features, and in this case there are 15 possible features it should give good results. Logistic Regression is very efficient, as well as simple to implement with fewer features.

### Performance Metrics

Aside from in-sample and out-sample errors, **precision**, **recall** and **F1-score** are used for assessing the performance of the four classifiers. A confusion matrix is an effective table layout that allows visualization of the performance of an algorithm as shown below.

		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Fig 3. confusion matrix

Measurements such as **precision**, **recall** and **F1-score** are derived from the confusion matrix, and the equations are listed here:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

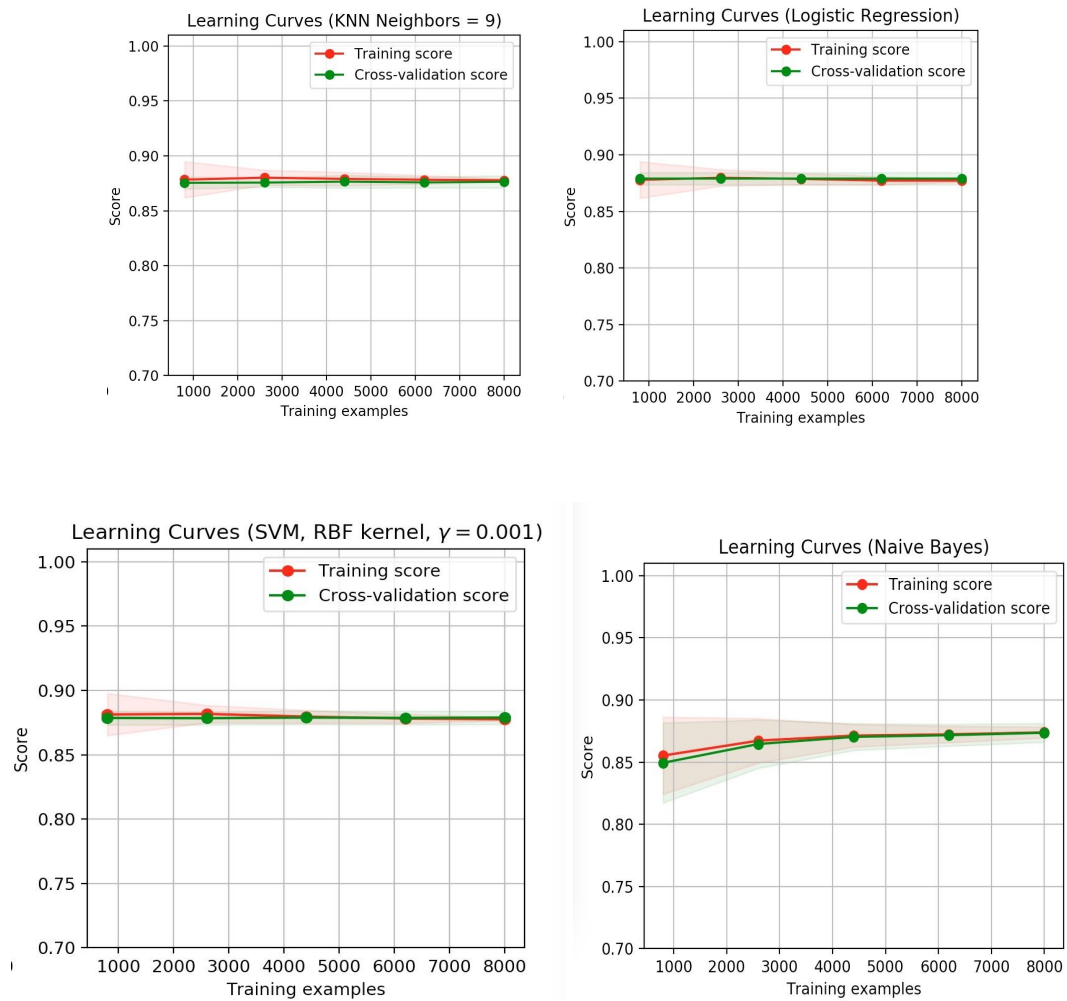
Precision will tell the percentage of true hit songs out of all the songs that we identified as hit songs. Recall will indicate the percentage of the hit songs we correctly predicted out of all the songs that are actually hit songs.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

F1 is a measure that is the harmonic mean of recall and precision, and ranges between 0 and 1. 1 means that we have perfect precision and recall, so we are aiming to optimize this value.

We chose these measures in consideration of the dataset. Because the dataset is so imbalanced (12%/88%), there is the possibility that the classifiers learn to only predict the majority class. Furthermore, if we were to use accuracy, then even simply predicting every single song to be not a hit song (-1) would result in an accuracy of approximately 88%. Thus, precision, recall, and F1 are the better measures than accuracy. For model complexity consideration, we plot the learning curves for the four models. Because of the imbalance of the dataset, the curves do not change a lot because of different sample sizes.

*Image 2*



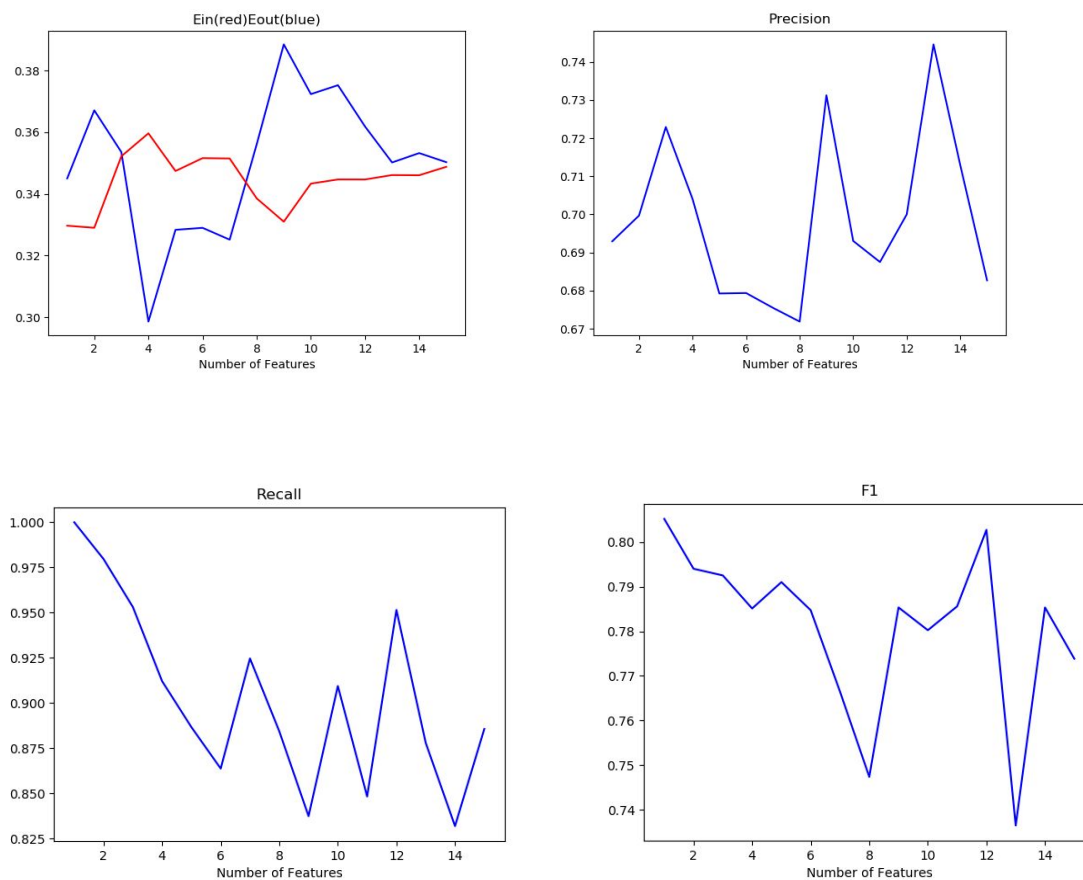
## RESULTS AND DISCUSSION

### Naive Bayes

Naive Bayes is a conditional probability model based on Bayes Theorem and the assumption of strong independence among the features. As the simplest model among the 4

classifiers, Naive Bayes gives us a baseline to determine the necessity/justification of using more complicated models. By applying the Naive Bayes on our resampled data of 1826 songs, (1224 hits and 612 non hit songs) we were able to calculate the average in-sample error rate, out-of-sample error rate, precision, recall and f1 among 100 trials with different K-feature values from 1 to 15.

*Image 3*



From the plots we see that both Ein and Eout show slight fluctuations through different K values but still maintain an accuracy around 30 to 38 percent which is not very accurate. For that reason we choose to focus more on precision, recall and F1 which show much higher accuracy.

Precision shows the greatest fluctuation as well as has the lowest accuracy among the 3 suggesting that Naive Bayes had the most trouble classifying the songs that were hits. However what is more interesting is that when K is equal to 1, recall has close to perfect accuracy which meant that the classifier was able to completely classify all of the hit songs correctly. The most influential factor in determining if a song was a hit was Time signature confidence.

Average results for Ein, Eout, Precision, Recall, F1 across all K

*Image 4*

```
Number of song predicted to be +1: 312
Number of song predicted to be -1: 56
ein: 0.3746183071525536
eout: 0.3531531531531531
precision: 0.7083333333333334
recall: 0.8565891472868217
F1: 0.7754385964912281
```

We then tested the trained Naive Bayes model on the entire, unscaled data set which contained only 1224 hit songs and over 8000 non-hits.

*Image 5*

```
ein: 0.3484304784988974
eout: 0.12239969799969808
precision: 0.1286253585719823
recall: 0.9228186274509802
f1: 0.2257719973179438
```

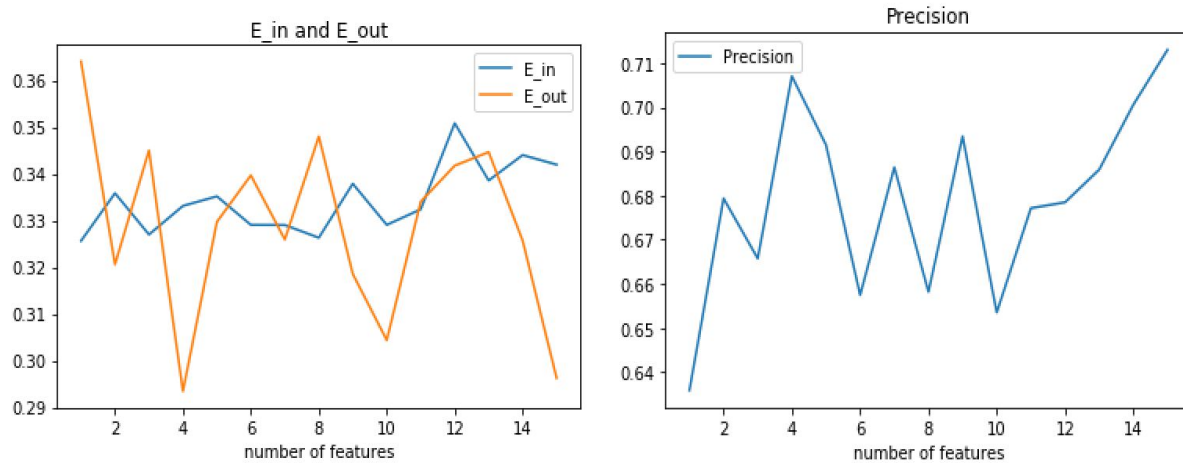
We see here that Ein roughly stays the same (because we are still using the balance dataset) however Eout, precision and F1 all drops drastically. However recall still remains similar to the results of the test on the balanced data suggesting that the model has captured the

majority of hit songs.

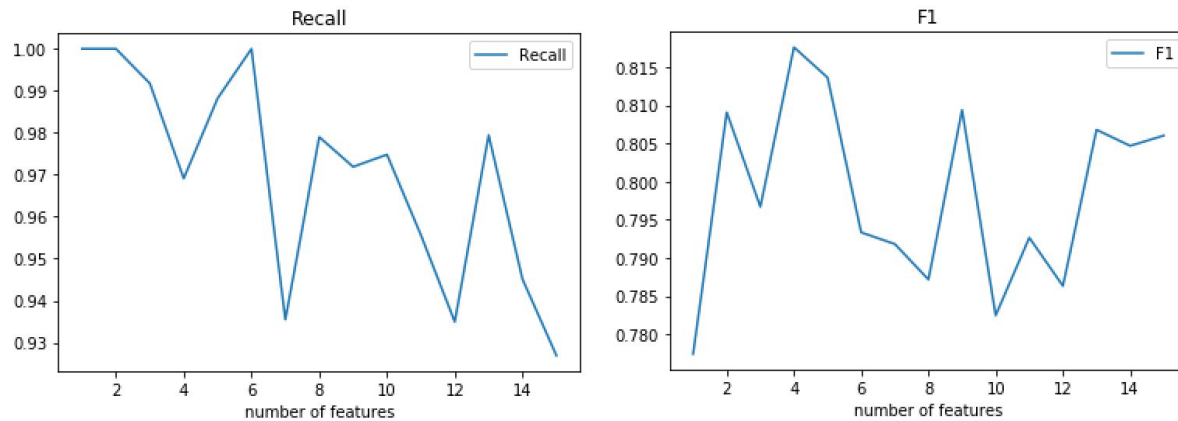
## Logistic Regression

Logistic Regression is a model using logistic function to get a probability for each prediction. We used the resample data (1836 songs), trained the model on 80% of the dataset and tested on the remaining 20%. After running the model hundreds of times for each number of features that were chosen by SelectKBest function, we then calculated the average of in-sample error rate, out-of-sample error rate, precision, recall and f1 . Moreover, among the five metrics, we utilized cross-validation with 10 folds to get more accurate results for  $E_{in}$  and  $E_{test}$  and used  $E_{test}$  to estimate  $E_{out}$ . The results are shown in image 6.

*Image 6*







We can see from the pictures that in-sample error rate and out-of-sample error rate still fluctuate around  $\frac{1}{3}$ , which implies the classifier predicts that all examples belong to the more frequent class as 67% labels in the resampling dataset are “-1”. However, precision, recall and f1 values are all very high in the results. Since it is still a little bit imbalanced, in order to get the optimal situation, instead of finding the minimum out-of-sample error rate, we looked at the number of features with the maximum f1 value, which should be four features here. The optimal results with 4 features are shown in image 7.

*Image 7*

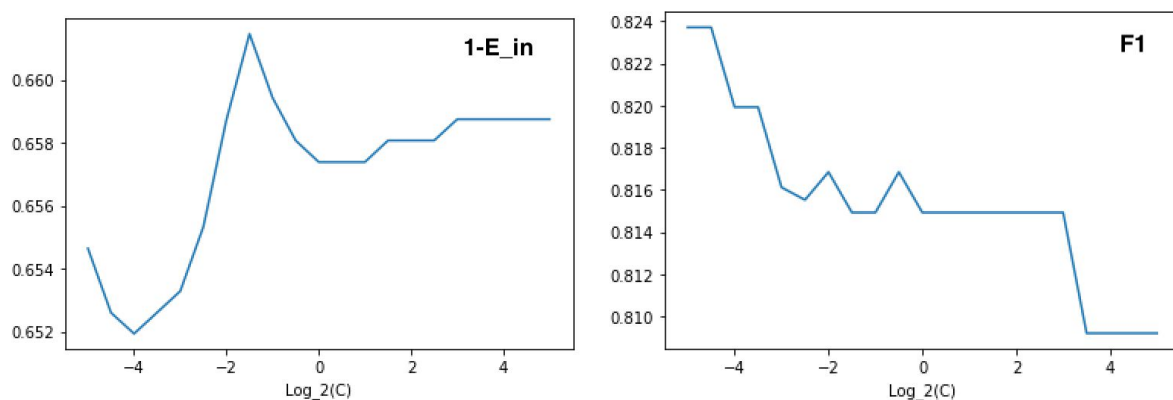
```
E_in: 0.3331003634330444
E_out: 0.29343629343629357
Precision: 0.7070422535211267
Recall: 0.9691119691119688
F1: 0.8175895765472312
```

Thus, it shows that with four features, we can obtain a precision of 70%, which means that 70% of the songs we predicted as hit songs are true hit songs and a recall of around 97%, which implies that 97% of true hit songs are successfully predicted as hit songs by the classifier.

Then, by `get_support` function, we get to know the four features are Artist Hotness, Key Confidence, Mode and Year.

Furthermore, we attempted to do regularization for this optimal 4 feature situation to restrict effective model complexity. The range of regularization constant was from 2 to the power of -5 to +5 in 0.5 increment each time. We plotted graphs of accuracy (1-in-sample error rate) and f1 for each regularization constant and attained the optimal situation with maximum f1 value. The results are shown in image 8.

*Image 8*



```
Log_2(C): -5.0
E_out: 0.30162399241346605
Precision: 0.7002724795640327
Recall: 1.0
F1: 0.8237179487179488
```

Compared to the results without regularization, we here get a higher recall value but the maximum f1 value does not change a lot. Thus, it seems that regularization does not have a clear

effect on the results. The reason why this would happen might be that the dataset we trained is imbalanced. Therefore, instead of being influenced by noise in the dataset, the classifier just tends to predict that all samples belong to the major class.

We then changed our training data and testing data. We made the resampling dataset as the training one and the whole original dataset as the testing one, and still chose the same 4 features in order to have a direct comparison. The results are shown below in image 9.

*Image 9 (No Regularization)*

```
E_in: 0.33821986990554354
E_out: 0.12239969799969799
Precision: 0.12398645181155703
Recall: 0.9869281045751631
F1: 0.22029725540257133
```

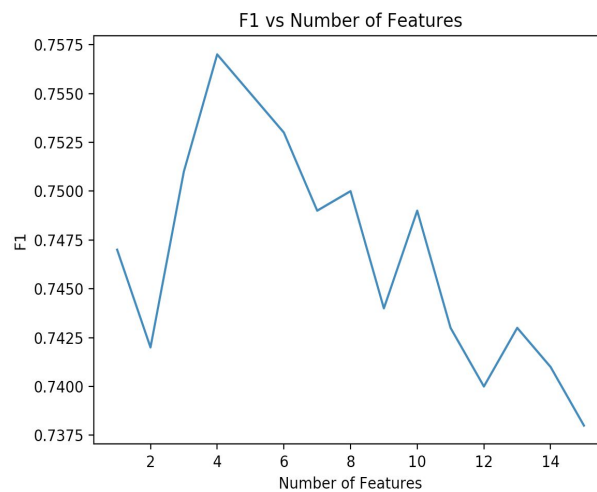
As we can see from the image, we have improved recall from around 97% to 99% which means that we successfully classified 99% of all the actual hit songs. However, precision and f1 drop greatly, which seems to be reasonable since we have a very small size of training data compared to the size of the testing data. The poor performance implies there may be some problems in our model. For example, the best 4 features we choose for the resampling dataset may not be the best 4 for the overall dataset. Also, a greater imbalance in the original one may have a large effect on the final results because we test on the original one to calculate precision, recall and f1. There is a lot of work we can do to improve our model in the future.

### K Nearest Neighbors

We used the SelectKBest function from scikit learn to determine the optimal number of

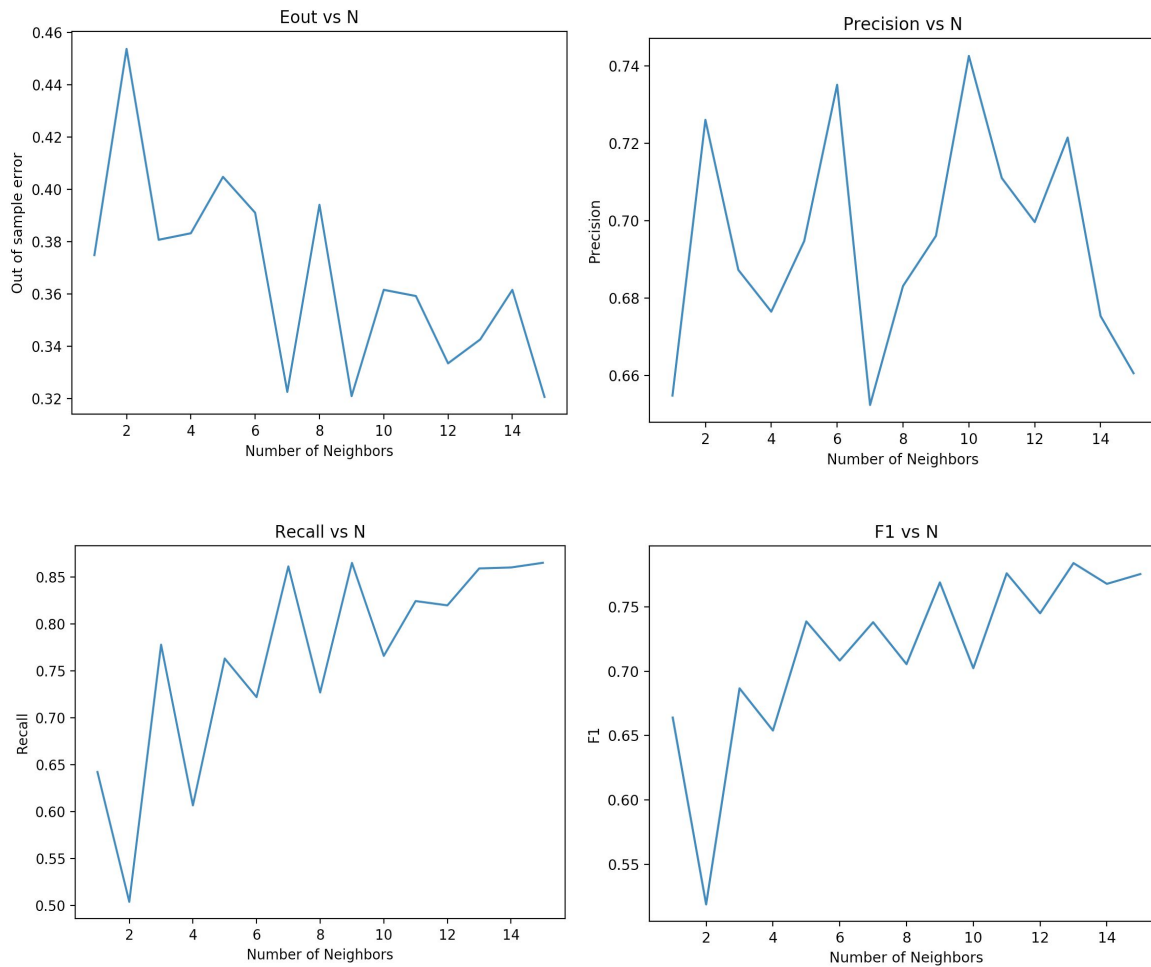
features, with `f_classif` as the scoring function. For KNN, the optimal number of features was 4, which were: Artist Hotness, Key Confidence, Mode, and Year. Below in image 3 we can see that F1 is highest when there are 4 features.

*Image 10*



Next, we determined the optimal number of neighbors for the KNN classifier. We ran each test 100 times, and took the average of out of sample error, precision, recall, and f1, shown below in image 11.

*Image 11*



We can see that the out of sample is lowest, recall, precision, and f1 are highest when there are 9 neighbors. Thus, when testing we went with 9 neighbors. For the resample data (1836 songs), we split it into 80% training data and 20% testing data by using scikit learn's `train_test_split` method. We trained on the training data, and then tested on the testing data and got the results shown below in image 12.

*Image 12*

```
Number of songs predicted to be +1: 308  
Number of songs predicted to be -1: 60  
ein: 0.36512856683865197  
eout: 0.36858608345187305  
precision: 0.6802569393003819  
recall: 0.8478102821104926  
f1: 0.7543238500460879
```

We obtained an out of sample error of 36.85%, a precision of 68.02%, a recall of 84.78%, and an f1 of 0.754. So, about 68% of what we predicted to be hit songs were songs that actually made it to the Billboard Hot 100's, and we successfully predicted approximately 85% of all hit songs with the KNN classifier.

We then tested our classifier on the original, imbalanced dataset. By training on the more balanced dataset we hope to have a better results classifying the imbalanced dataset, and hopefully the classifier will not have the issue of classifying everything as -1. The results are shown below in image 13.

*Image 13*

```
Number of songs predicted to be +1: 7975  
Number of songs predicted to be -1: 2025  
ein: 0.3763869090045141  
eout: 0.1252000000000001  
precision: 0.1334169278996865  
recall: 0.869281045751634  
f1: 0.2313294923361235
```

As we can see, the performance of the classifier has dropped immensely. Precision is at 13.34%, so out of everything we classified as a hit song only approximately 13% of them made it to the Billboard Hot 100's. Recall increased to 86.93%, so we successfully classified 87% of all the actual hit songs, and f1 lowered from .75 to .23. However, while the performance is not good here, we can see that we predicted 79.75% of songs to be hit songs, and 20.25% not hit songs. This is an improvement from before, where for the imbalanced dataset we predicted 99.1% of them to be not hit songs. So by training on the resampled data did help the classifier in determining whether or not a song will make it to Billboard's Hot 100. However, considering the poor performance, there is probably an issue with the features we are using. Either the feature selection could be better, or the features themselves are not indicative of whether or not a song will be a hit song.

### Support Vector Machines

SVM is chosen because it is widely used for classification tasks, including the binary classification in our case, and it has the advantage to avoid overfitting. SVM algorithm aims to find a hyperplane in an N-dimensional space ( N means the number of features ) that classifies

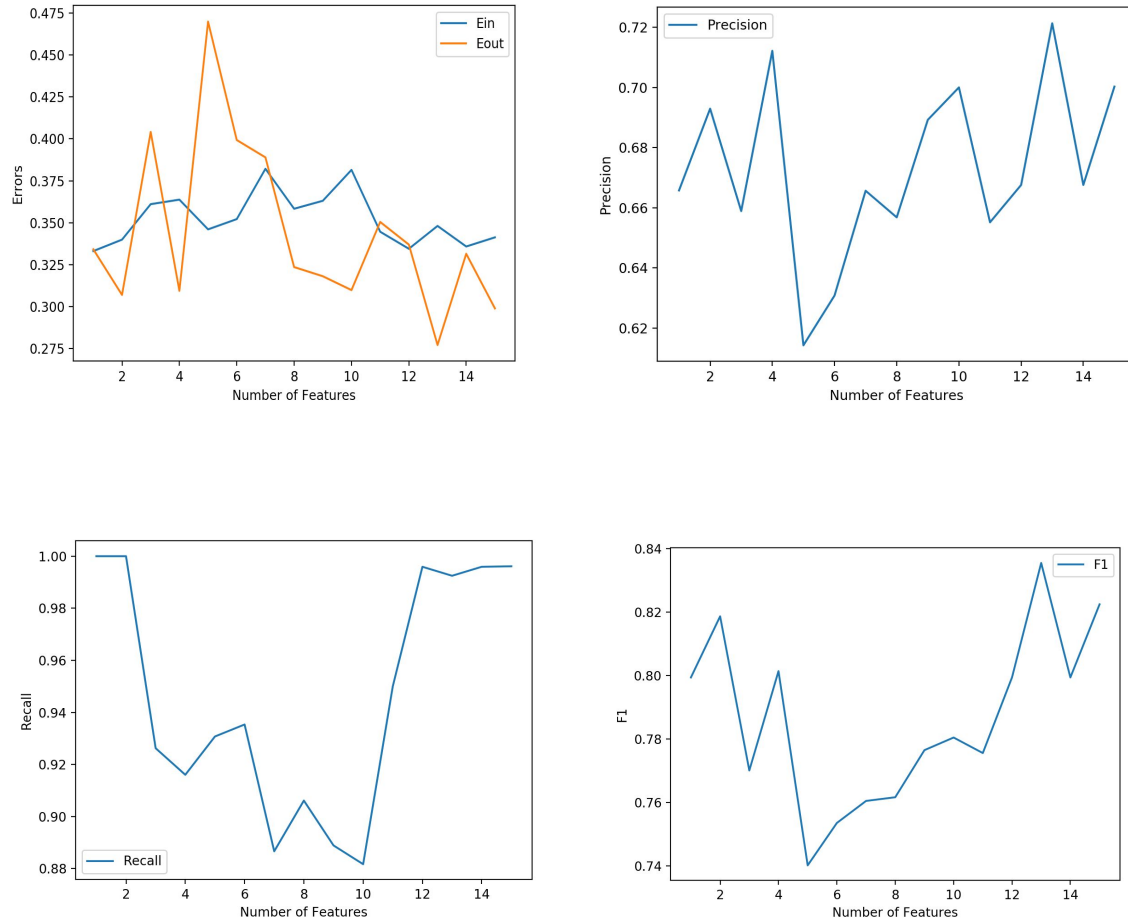
the data points. First, we resampled the original dataset to 1224 cases of +1 labels and 612 cases of -1 labels. Then, we split the resampled data with 0.8 training size and 0.2 testing size. We used cross validation score with cv=10 to in order to get more accurate Ein and Eout. After 100 runs, we calculated the average Ein, which was 1 - cross\_validation\_score of resampled dataset and resampled labels, average Eout, which was 1 - cross\_validation\_score of original dataset and original labels. Since our dataset has imbalanced two classes, Eout can not be viewed as an accurate indicator for the result. So, we also included precision, recall, and F1 scores as Eout was no longer effective. The result showed that SVM got highest F1, highest precision, a very high recall, and the lowest Eout when there were 13 features. The 13 features were: artist hotness, artist familiarity, danceability, duration, end of fade in, energy, key, key confidence, loudness, mode, start of fade out, tempo, and year. Here is the result:

*Image 14*

```
E_in: 0.3481
E_out: 0.2771
Precision: 0.7213
Recall: 0.9924
F1: 0.8354
```



Image 15



Also, in the optimal case of 13 features, we trained on the whole resampled data, and tested on the original dataset. This time, precision and F1 decreases a lot. Yet, Eout also decreases, and recall is still very high. Since high recall means that our model returned most of the relevant results, recall value of 1.0 meant that our model returned most of the important

results.

*Image 16*

```
E_in: 0.33333434467967593
E_out: 0.12239969799969792
Precision: 0.13079717888437697
Recall: 1.0
F1: 0.23133623133623135
```

In addition, we make hyperparameters selection using grid search with cross-validation in the optimal case of 13 features. We resample the original dataset by randomly selecting 100 data samples with -1 label and 200 data samples with +1 label. We compare rbf kernel and linear kernel, 0.001 gamma value and 0.0001 gamma value, degree constant of 1,2, and 4, and regularization constant C of 0.5, 1, 2, and 10. Best parameters set found on the original dataset is {'C': 2, 'degree': 1, 'gamma': 0.001, 'kernel': 'rbf'}. Below is the report for precision, recall, and f1:

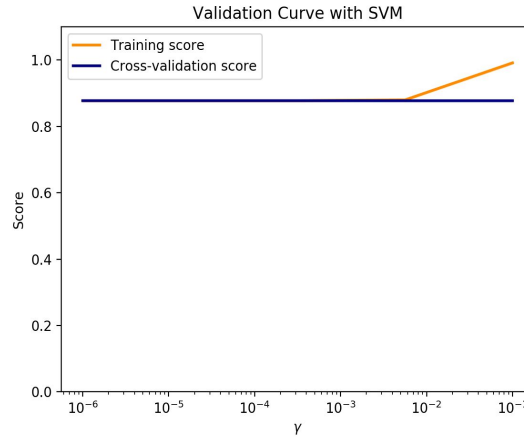
*Image 17*

Detailed classification report:

The model is trained on the full development set.  
The scores are computed on the full evaluation set.

	precision	recall	f1-score	support
-1	0.50	0.33	0.40	3
1	0.67	0.80	0.73	5
accuracy			0.62	8
macro avg	0.58	0.57	0.56	8
weighted avg	0.60	0.62	0.60	8

Image 18



We also tested training score and validation score for SVM in terms of gamma. Because of imbalance of our dataset, the scores are not affected by the gamma values. In the future, we will graph model complexity curves in terms of F1.

## CONCLUSIONS

Table 2: Performance of all 4 classifiers over the resampled data

Classifier	Ein	Eout	Precision	Recall	F1
KNN	.3762	.3782	.6802	.8318	.7480
SVM	.3481	.2771	.7213	.9924	.8354
NB	.3746	.3532	.7083	.8566	.7754
LR	.3452	.3016	.7003	1.0000	.8237

*Table 3: Performance of all 4 classifiers over the original data set, after training on the resampled data*

Classifier	Ein	Eout	Precision	Recall	F1
KNN	.3764	.1252	.1334	.8693	.2313
SVM	.3333	.1224	.1308	1.0	.2313
NB	.3484	.1224	.1286	.9228	.2258
LR	.3382	.1224	.1240	.9869	.2203

As we can see from tables 3 and 4, the results for testing on a 20% split of the resampled data yielded better results. We did a much better job successfully classifying all the hit songs, as well as being correct with our prediction of hit songs. Comparatively, testing on the full imbalanced dataset gave terrible results other than the resample. But as we can see, we are predicting that the majority of the songs are hit songs that results in a high recall, but the precision suffers because of it.

Our original goal was to make a classifier that could help artists determine whether or not a song they were making would be a hit song early on in the process, as well as determine the features that were important in making a hit song. The results showed that the classifier does not perform well enough to perform the task that we meant for it, and the features that were most important were different across each classifier. This means that we still need more work on selecting and comparing features.

Also, since the hypothesis is that the precision will make our models better for training,

in the future, we will research on whether the precision of the ranking ( e.g. 10% rank ) on the Billboard make the result better. We will also graph performance graphs in terms of F1, precision, and recall instead of cross validation scores in order to get a more accurate observation of whether model complexity is enough.

We will use Naive Bayes, a simple model, select the best feature returned by Naive Bayes, set its result as a benchmark, and compare whether our results of other models ( complicated models like Logistic Regression and SVM ) are better than the benchmark.

Furthermore, when determining the K best features of the dataset via the selectKBest method from sklearn, the features selected differed between the training data and the testing data, which contributed negatively to the results. In the future, we should spend more time on setting the features so that they are the same for both the resampling dataset and the original one, and we expect to see improvements in the results.

In addition, having only 1% of the Million Song Dataset definitely was a big disadvantage. There is so much less data to train on, and the results we obtained may have suffered a lot as a result. Also, the smaller dataset we downloaded is greatly imbalanced, which is one of the reasons why our performances are not that ideal. If possible, downloading the full one million songs (300 gigabytes), dealing with imbalanced problems and using that to train and test would give better results.

Lastly, the feature selection process was not optimal. We could use some data mining techniques to improve feature selection, hopefully increasing the results. However, there is always the probability that the features in our data set are not suited towards predicting hit songs,

thus we could look for more data with different features.

## REFERENCES

1. Abu-Mostafa, Yaser S et al. Learning From Data. AML, 2012.
2. Borg, Nicholas. Hokkanen, George. "What Makes For A Hit Pop Song? What Makes For A Pop Song". Cs229.Stanford.Edu, 2019,  
<http://cs229.stanford.edu/proj2011/BorgHokkanen-WhatMakesForAHitPopSong.pdf>.
3. "How Accurately Can You Forecast A Song's Hit Potential? — Hyperlive". Hyperlive, 2019,  
<https://hyperlive.fm/hyperlive-blog/2018/1/27/how-accurately-can-you-forecast-a-songs-hit-potential>.
4. <http://millionsongdataset.com/pages/additional-datasets/>
5. Sanchez, Daniel. "What Streaming Music Services Pay (Updated For 2019)". Digital Music News, 2019,  
<https://www.digitalmusicnews.com/2018/12/25/streaming-music-services-pay-2019/>.

6. Reiman, Minna. "Predicting Hit Songs With Machine Learning". EXAMENSARBETE INOM TEKNIK, GRUNDNIVÅ, 15 HP STOCKHOLM, SVERIGE 2018, 2019, Accessed 7 Dec 2019.
7. Nasreldin, Mohamed. "Song Popularity Predictor". Towardsdatascience.Com, 2019, <https://towardsdatascience.com/song-popularity-predictor-1ef69735e380>.
8. Pachet, Francois & Roy, Pierre. (2008). Hit Song Science Is Not Yet a Science.. 355-360.
9. UC Davis, Poverty Center. "What Are The Annual Earnings For A Full-Time Minimum Wage Worker? - UC Davis Center For Poverty Research". UC Davis Center For Poverty Research, 2019, <https://poverty.ucdavis.edu/faq/what-are-annual-earnings-full-time-minimum-wage-worker>.
10. Newcomb, Alyssa. "Spotify Trounces Apple Music In Competition For Streaming Music Service Paid Subscribers". Fortune, 2019, <https://fortune.com/2019/02/06/spotify-apple-music-paid-subscribers-streaming-music-service/>.
11. Nick Dupoux, Angela Xue. "Predicting A Song's Commercial Success Based On Lyrics

And Other Metrics". Cs229.Stanford.Edu, 2019,

<http://cs229.stanford.edu/proj2014/Angela%20Xue,%20Nick%20Dupoux,%20Predicting%20the%20C>

12. Wang, Kedao. "Predicting Hit Songs With MIDI Features". Cs229.Stanford.Edu, 2019,

<http://cs229.stanford.edu/proj2014/Kedao%20Wang,%20Predicting%20Hit%20Songs%20with%20MIDI%20Musical%20Features.pdf>

13. Yun, Lang-Chi, Yang, Yi-Hsuan, Hung, Yun-Ning, Chen, Yi-An. "HIT SONG PREDICTION FOR POP MUSIC BY SIAMSE CNN WITH RANKING LOSS".

Stat.ML, 2019, Accessed 7 Dec 2019.