Introduction/Background

CS4641 Project Proposal - Group 89

1. Introduction/Background

Our project aims to predict whether or not a song has what it takes to be a “hit”. Using metrics like length, tempo, artist, hook, bridges, and breaks etc we will find a true rating for what makes a popular song, popular. Our data set contains a list of ~40,000 songs along with these metrics that will be useful for determing a “hit” rating. A study conducted by Claremont Graduate University used AI to analyze brain response to music and achieved 97% accuracy in predicting hit songs [1].

2. Problem Definition & Motivation

In the world of music, understanding what makes a song a hit is a complex challenge. With millions of songs released every year, determining which ones will become popular or go viral has traditionally been unpredictable. Our project tackles this problem by leveraging machine learning to predict whether a song will become a hit. While there is no magic formula to guarantee that a song will become a hit, we are going to analyze which features are most crucial for a song becoming a hit [2]. This project is valuable not just for the music industry, but for artists and producers who want insights into the factors driving popularity. It could also help streaming platforms and record labels identify potential hits early, enabling better promotional strategies.

3. Methods + Libraries

We plan to clean up our dataset by normalizing and handling missing data using Pandas [3]. We will be using Scikit-Learn to select which features are the most important to consider using recursive feature elimination to eliminate irrelevant and weak features as well reduce the dimensionality using principal component analysis by transforming the features into a set of linearly uncorrelated components while retaining the maximum variance in the data [3].

We started by pre-procesing our data. We used a combination of forward feature selection and

backward feature selection, and we decided to keep any feature selected by either of those methods to give us our initial eighteen features. Additionally we have two seperate responses: one as defined by our original datatset (a binary representation of either a hit or a flop), and one pulled directly from the spotify API (measured as a popularity value out of 100). We used the predefined hit or flop values to conduct our preliminary methods.

The first method we implemented was Logistic Regression because it is well-suited for binary classification problems. The next method we used was Random Forest Classifier because it has proven to be one of the more effective methods of determinining whether a song will be a hit or not, which was shown through our implementation as well [3]. In order to implement both of these methods, we used the Scikit-Learn libraries. Additionally we impleneted a Neural Network through Keras using TensorFlow and testing mutltiple different model architectures until landing on one that was able to predict hits and flops quite well.

Finally, we conducted PCA on our current features as well as recursive feature elimination (RFE) to pre-process our data, and re-ran this processed daa through our logistic regression and random forrest to produce the best possible combination. We did not perform PCA before our Neural Network method as we did not expect that make a big impact. From we were able to test multiple combinations of pre-processing methods and machine learning algorithms.

4. Results and Discussion

Quantitative Metrics

- Determine on a scale of 1-100 how likely a song is to be considered a hit, with 100 being most likely to be a hit and 1 to be least likely
- Binary “Hit” or “Not Hit” if the scale rating is meets a certain threshold (~70)
- Estimate the number of stream within a certain time period based off the calculated score

Project Goals

- Be able to determine with relative accuracy whether or not a song will be a hit.
- Our scale rating should have some correlation to the number of streams (popularity) or the estimated popularity over time

Results

Logistic Regression

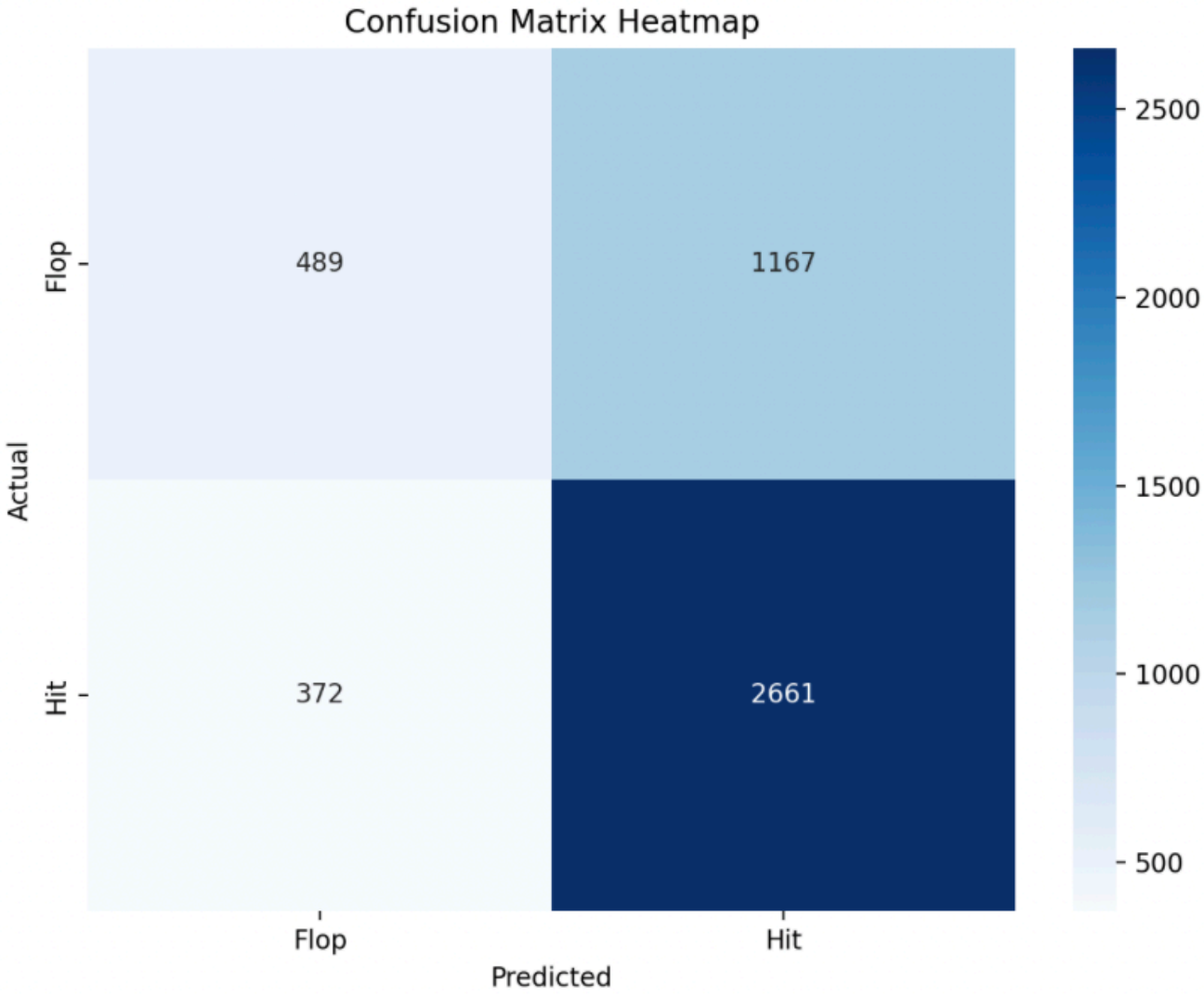
On average, we were able to achieve around ~67% accuracy which means it was able to capture some of the predictive patterns in the data however, it was not quite as high as we had originally hoped. In terms of the quantitative metrics we used to measure our model, outside of the accuracy score we kept track of a confusion matrix to count of true positives, true negatives, false positives, and false negatives where any large number of false positives or false negatives could be interpreted as our model missing some sort of key patterns

```
Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.30      0.39      1656
           1       0.70      0.88      0.78      3033

    accuracy                           0.67      4689
   macro avg       0.63      0.59      0.58      4689
weighted avg       0.65      0.67      0.64      4689

Accuracy Score: 0.6717850287907869
```

As far as explaining why we had a lower than hoped accuracy there are a few potential reasons. The basis of logistic regression is that it is a linear model, meaning it can lack it in capturing the true complex nature of non-linear relationships in the data. There is obviously an intricate relationship between different features like tempo, valence, loudness etc that logistic regression just simply can't model in an efficient way because of its non linear nature. Further, our dataset could have introduced a bit of noise, for example you can consider metrics like "danceability" or "valence" as subjective in a sense which could skew our ability to predict a hit song. It might also be nonoptimal to include features that might not have as much of a direct impact for determining whether a song is a hit or not. Seen by things like "chorus hit time" or "sections." I think the first step in improving our model could be through redefining what we consider exactly a hit to be.

As we used a pre-made dataset with this already created for us, we are currently working on querying the spotify api as they have a metric there called popularity. A track's popularity is

between 0 and 100, with 100 being the most popular. The popularity is calculated by a spotify algorithm and is based mainly on the total number of plays the track has had and how recent those plays are. Incorporating this into our dataset and making a threshold for what is a hit (say a popularity score >= 80) we could fine tune our dataset and potentially get a higher accuracy score.



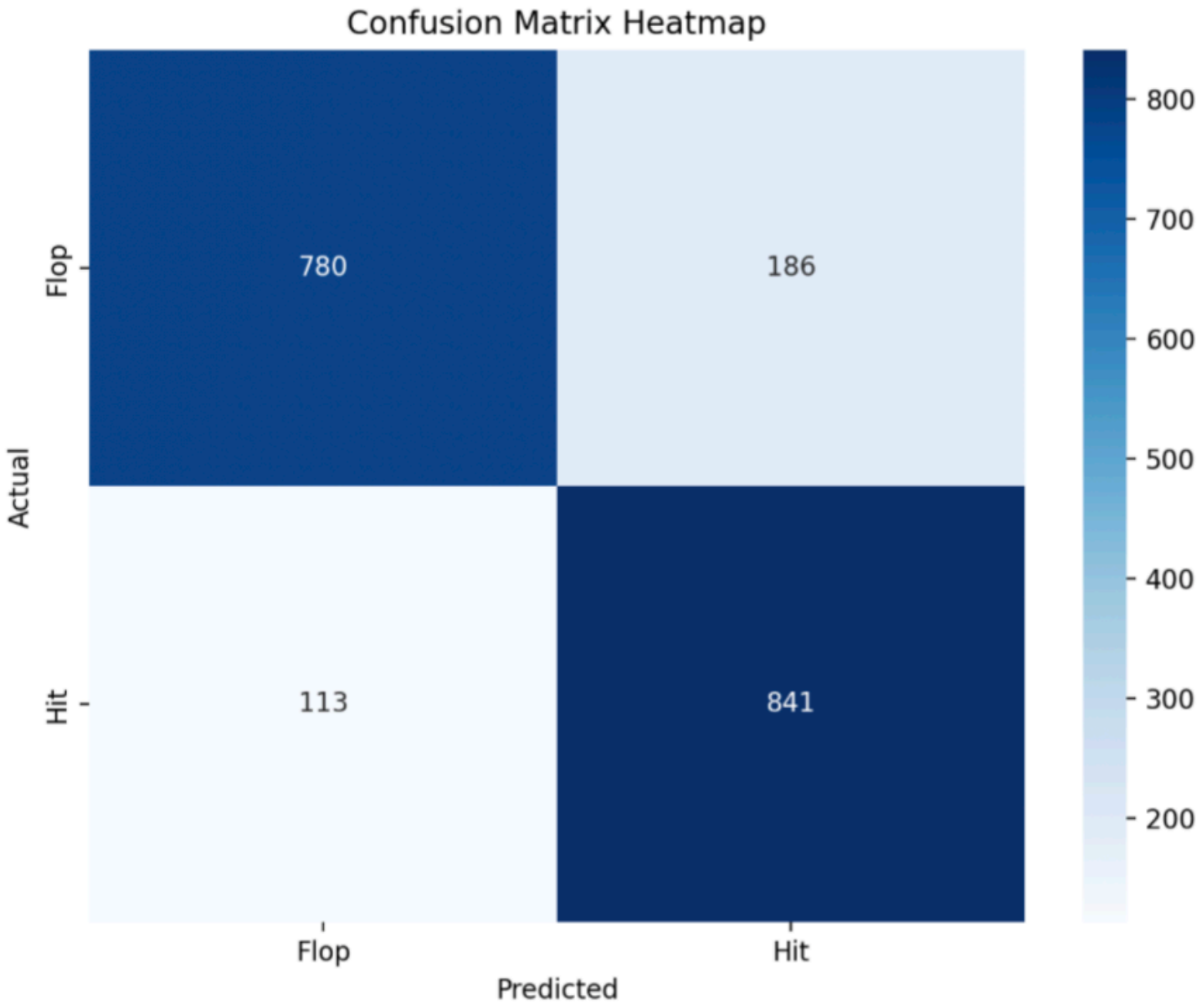
This confusion matrix shows the breakdown of correct and incorrect predictions made by the model. Each square block in the matrix represents the count of either true positives, false positives, true negatives, and false negatives. High values along the diagonal indicate accurate classifications of both hits and flops, while off-diagonal values highlight misclassifications, suggesting where the model struggled to distinguish between hits and flops accurately. The model tends to perform better in identifying hits correctly (2661 True Positives) but has a high rate of False Positives as well (1167), indicating it often mistakenly classifies flops as hits meaning we should rethink how our dataset is derived in general.

Random Forest

Using the Random Forest method we were able to achieve roughly 84% accuracy which means it was able to capture a good amount of the predictive patterns in the data.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.81      0.84       966
           1       0.82      0.88      0.85       954

    accuracy                           0.84      1920
   macro avg       0.85      0.84      0.84      1920
weighted avg       0.85      0.84      0.84      1920

Accuracy: 0.8442708333333333
```

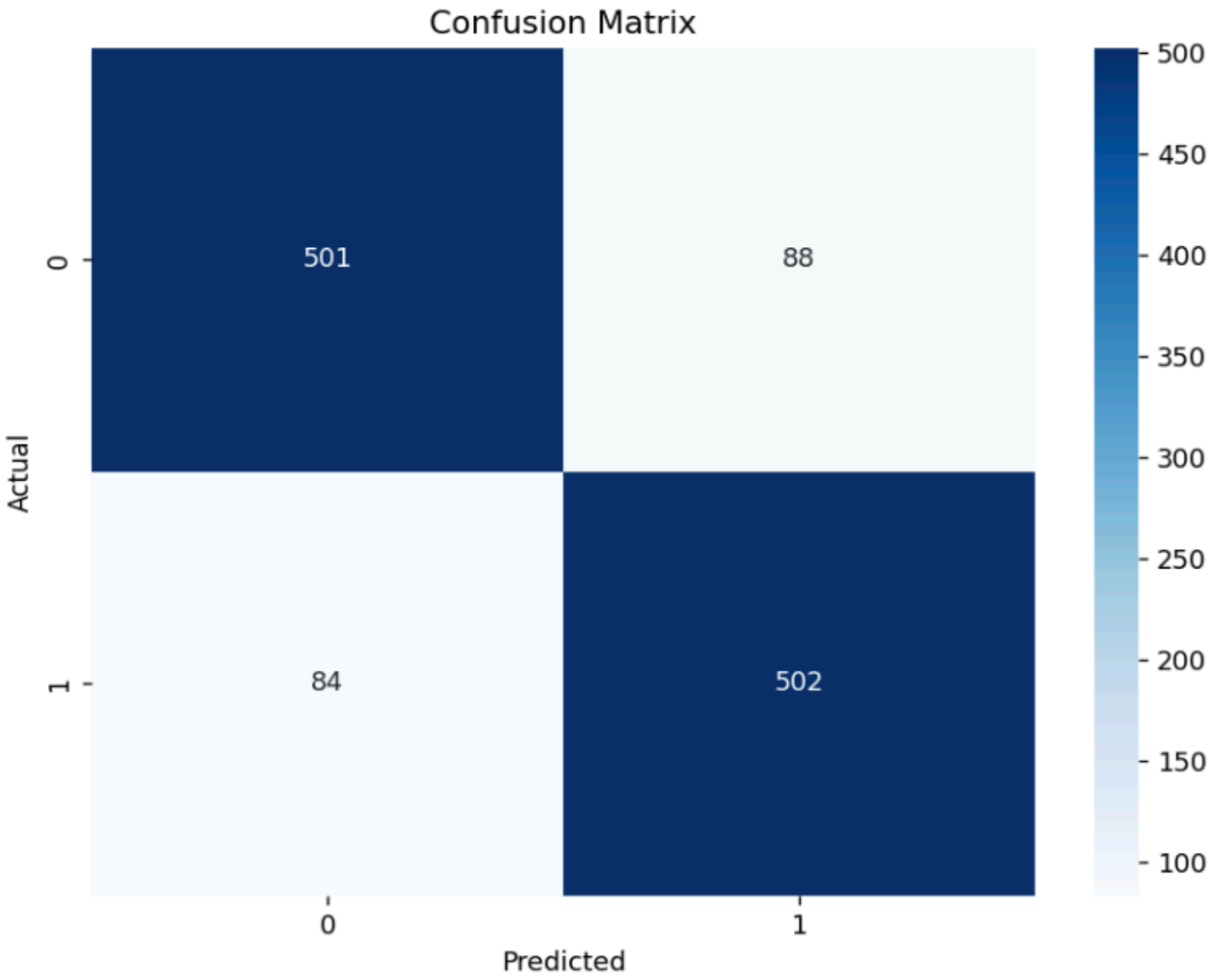
This was better than the linear regression method but there is still room for improvement to find a model that has a higher accuracy when predicting whether a song will be a hit or a miss.

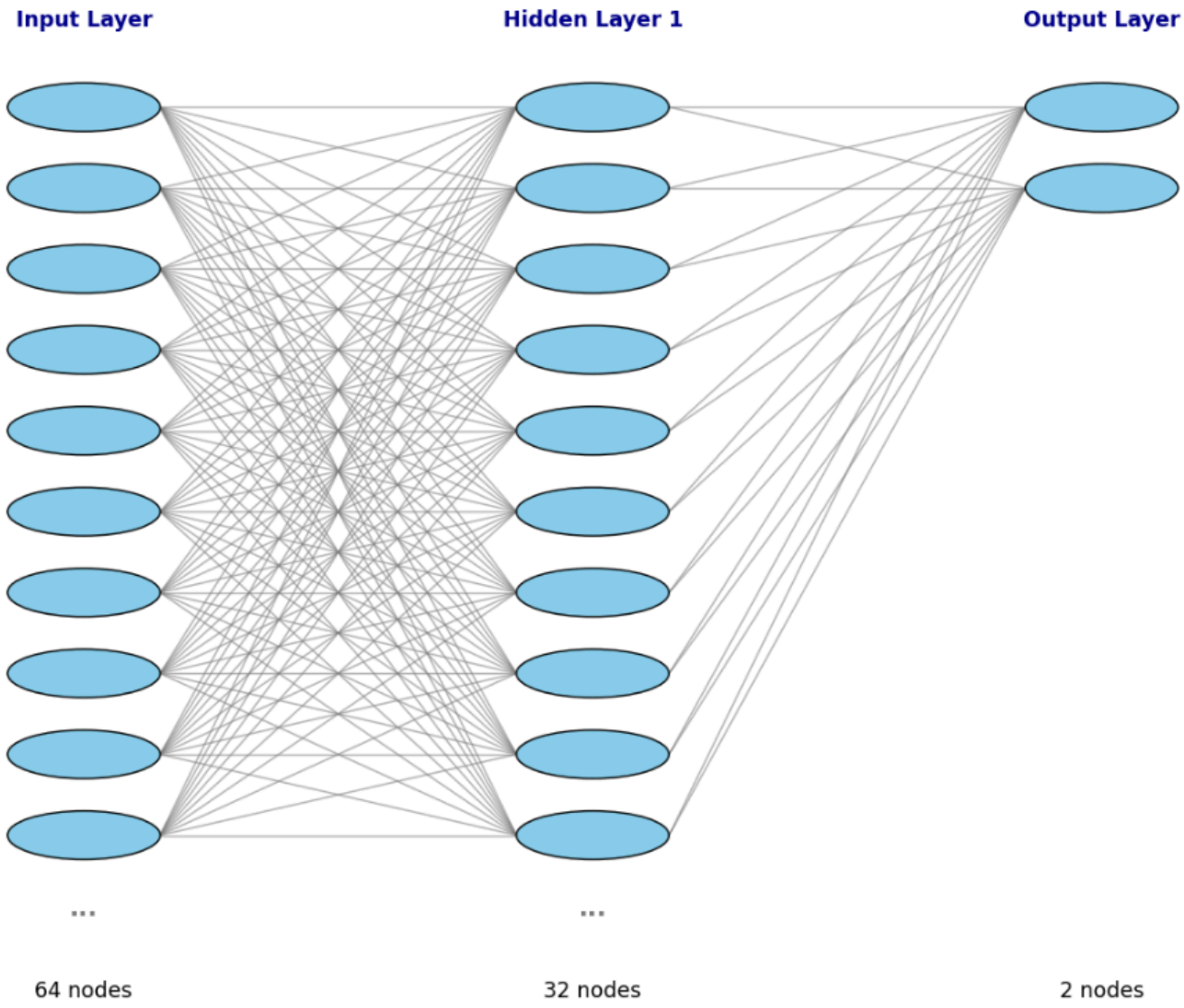


The confusion matrix shows that the model works best at correctly predicting a hit song but still does a very good job at predicting flops, especailly compared to the logistic regression method which did a poor job of properly predicting flops.

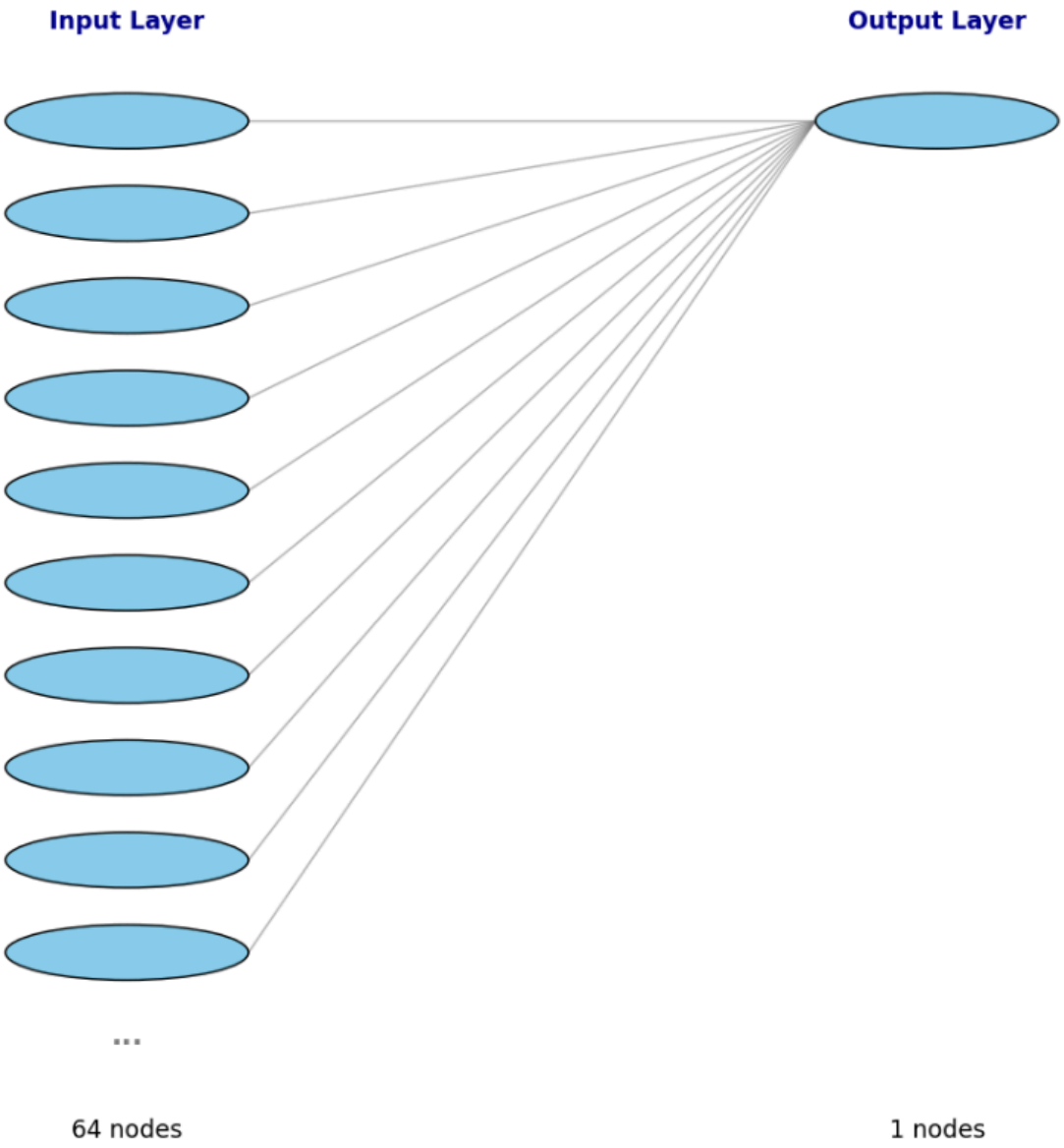
Neural Network

The neural network method proved to be very effective at taking our hit song data and accurately predicting whether or not a song would be a hit or flop. Our test accuracy was .85 for this model and the confusion matrix is shown below. The structure for this method is also shown below the confusion matrix. Our first attempt features the input layer, with ReLu as the activation function followed by one hidden layer also utilizing ReLu. Finally we used Softmax for our classification method and Sparse Categorical Cross Entropy as our loss function.


Confusion Matrix
501
88
84
502
0
1
Actual
0
1
Predicted
500
450
400
350
300
250
200
150
100



We then tried multiple different architectures for our neural network hoping to improve our accuracy, but were unable to find one that performed better than 85% accuracy. We did however find a simpler one that also performed at this benchmark of .85 accuracy. The structure is shown below, and features just the inputs passed through the swish activation function. Followed by sigmoid as the classification method and binary classification as the loss function. This proved to be the overall best method out of all that we tested.



Next Steps

To improve our project, our next steps would involve refining the dataset by incorporating Spotify's popularity metric, which provides a clearer industry-standard definition of a "hit." By setting a threshold (something like popularity ≥ 80) and creating other derived features which are inherently apart of the songs themselves, such as tempo-to-duration ratio or normalized valence. This will in theory better capture the relationships between features. Additionally, we will clean noisy, subjective features like "danceability" and "valence" using clustering or outlier detection. On the modeling side, we could plan to optimize hyperparameters through Grid Search or Random Search and explore ensemble methods like boosting or stacking to enhance prediction accuracy.

Further, we intend to focus on error analysis, particularly reducing False Positives and False

Negatives by identifying biases in the data and model predictions. Employing things like K-Fold Cross-Validation will ensure our evaluation works as intended. Further, some of the potential applications for this project are pretty wide, ranging from helping streaming platforms and record labels identify hit songs early, to offering insights for artists and producers about elements that drive popularity. Additionally, the model could be integrated into recommendation systems for playlist curation or targeted advertising.

5. References

[1]S. Kapoor, A. Narayanan, “No, you still cannot predict hit songs using machine learning,” reproducible.cs.princeton.edu. https://reproducible.cs.princeton.edu/predicting-hits.html

[2]I. Dimolitsas, S. Kantarelis, and A. Fouka, “SpotHitPy: A Study For ML-Based Song Hit Prediction Using Spotify.” Available: https://arxiv.org/pdf/2301.07978

[3]F. Khan et al., “Effect of Feature Selection on the Accuracy of Music Popularity Classification Using Machine Learning Algorithms,” Electronics, vol. 11, no. 21, p. 3518, Oct. 2022, doi: https://doi.org/10.3390/electronics11213518.

[4]E. Georgieva, M. Suta, and N. Burton, “HITPREDICT: PREDICTING HIT SONGS USING SPOTIFY DATA STANFORD COMPUTER SCIENCE 229: MACHINE LEARNING.” Available: https://ccrma.stanford.edu/~egeorgie/documents/HitPredict_Final.pdf

Gantt Chart

GANTT CHART

PROJECT TITLE	Spotify Hit Predictor

TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION
Project Proposal				
Introduction & Background	All	9/23/2024	9/30/2024	7
Problem Definition	All	9/23/2024	9/30/2024	7
Methods	All	9/23/2024	9/30/2024	7
Potential Results & Discussion	All	9/23/2024	9/30/2024	7
Video Recording	Beyonce, Desirae	9/30/2024	10/4/2024	4
GitHub Page	Scott, Charles	9/30/2024	10/4/2024	4
Data Collection / Processing				
Data Sourcing	Charles, Samuel	10/5/2024	10/10/2024	5
Data Cleaning	Charles, Samuel	10/8/2024	10/12/2024	4
Model 1				
Model Selection	Desirae, Beyonce	10/10/2024	10/14/2024	4
Data Pre-Processing	Charles, Samuel	10/14/2024	10/17/2024	3
Model Coding	Scott, Beyonce, Desirae	10/17/2024	10/27/2024	10
Results Evaluation and Analysis	Scott, Beyonce, Desirae	10/25/2024	10/31/2024	6
Midterm Report	All	10/31/2024	11/7/2024	7
Model 2				
Model Selection	Scott, Charles, Samuel	10/10/2024	10/14/2024	4
Data Pre-Processing	Charles, Samuel	10/17/2024	10/24/2024	7
Model Coding	Charles, Samuel	10/24/2024	11/5/2024	11
Results Evaluation and Analysis	Charles, Samuel	11/8/2024	11/14/2024	6
Model 3				
Model Selection	Scott, Charles, Samuel	10/10/2024	10/14/2024	4
Data Pre-Processing	Charles, Samuel	10/17/2024	10/24/2024	7
Model Coding	Scott, Beyonce, Desirae	10/31/2024	11/10/2024	10
Results Evaluation and Analysis	Scott, Beyonce, Desirae	11/10/2024	11/16/2024	6
Evaluation				
Model Comparison	All	11/16/2024	11/20/2024	4
Presentation	All	11/20/2024	11/25/2024	5
Recording	All	11/25/2024	11/26/2024	1
Final Report	All	11/20/2024	12/1/2024	11

Link to Full Gantt Chart

Contributions

Name	Project Proposal

Charles Labee	Project Research Gantt Chart Project Webpage Creation
Scott Lenney	Project Research Proposal Write-Up Project Webpage Creation
Beryonce McCrary	Project Research Video Recording
Desirae Serrano-Diaz	Project Research Video Recording
Samuel Walls	Project Research Proposal Write-Up Project Webpage Creation

---

CS4641_proj_website is maintained by slenney3.
This page was generated by GitHub Pages.