**FIRST CHUNK OF EACH SONG (OLD CHUNKING METHOD IN V5 BASELINE CODE)**

* learning rate = 5e-5, epochs = 5, batch\_size=8, train\_size = actual size

Text

Description automatically generated with medium confidence

* learning rate = 5e-5, epochs = 5, batch\_size=8, layers=0, train\_size = actual size
* learning rate = 5e-5, epochs = 5, batch\_size=8, layers=1, train\_size = actual size

A picture containing text

Description automatically generated

* learning rate = 5e-5, epochs = 5, batch\_size=8, layers=1, dropout=0.1, train\_size = actual size (model\_final\_v3dropoutWithOneLayer)

A picture containing text

Description automatically generated

Chart, treemap chart

Description automatically generated Chart, line chart

Description automatically generated

Graphical user interface, text, application

Description automatically generated

* How to improve accuracy results:
  + Add more data
  + Add more layers – 1-12 layers
  + Change dropout rate – 0.1, 0.3, 0.5
  + Add more epochs – 5, 10, 15
  + Change learning rate – 5e-5, 5e-6, 5e-3

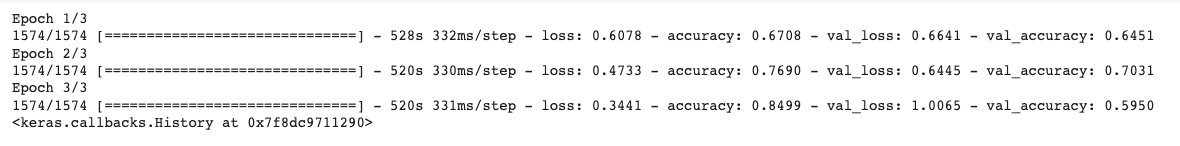
**NEW CHUNK CODE IN V6**

* New chunk code – first 1000 songs in train, first 300 songs in test
  + Train layers 1, learning rate 5e-5, dropout 0.1

A picture containing text

Description automatically generated

* New chunk code – **Full dataset**
  + Train layers 1, learning rate 5e-5, dropout 0.1, epochs=3



* + WITHOUT AGGREGATION RESULTS

Chart, line chart

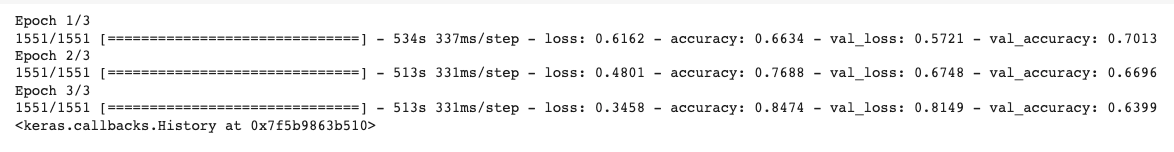
Description automatically generated

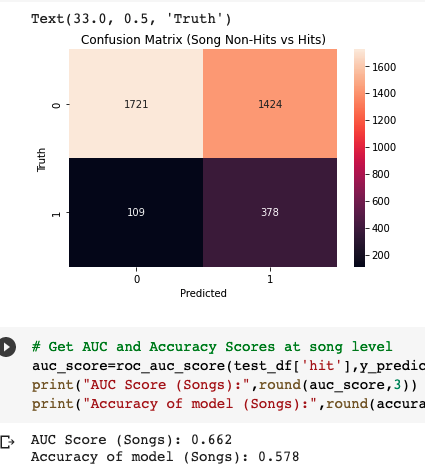
Chart, treemap chart

Description automatically generated

—----------------------------------------------------------------------------------------------------------------------

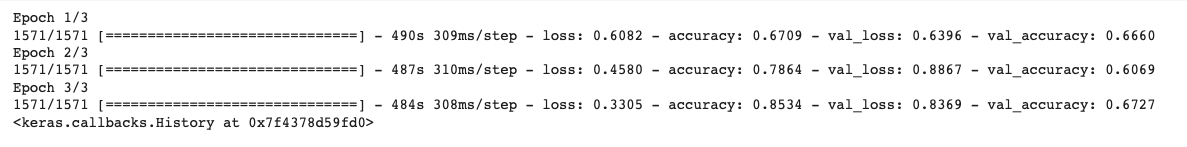
* New chunk/prob\_aggregation code – **Full dataset (RUN 2)**
  + Train layers 1, learning rate 5e-5, dropout 0.1, epochs=3



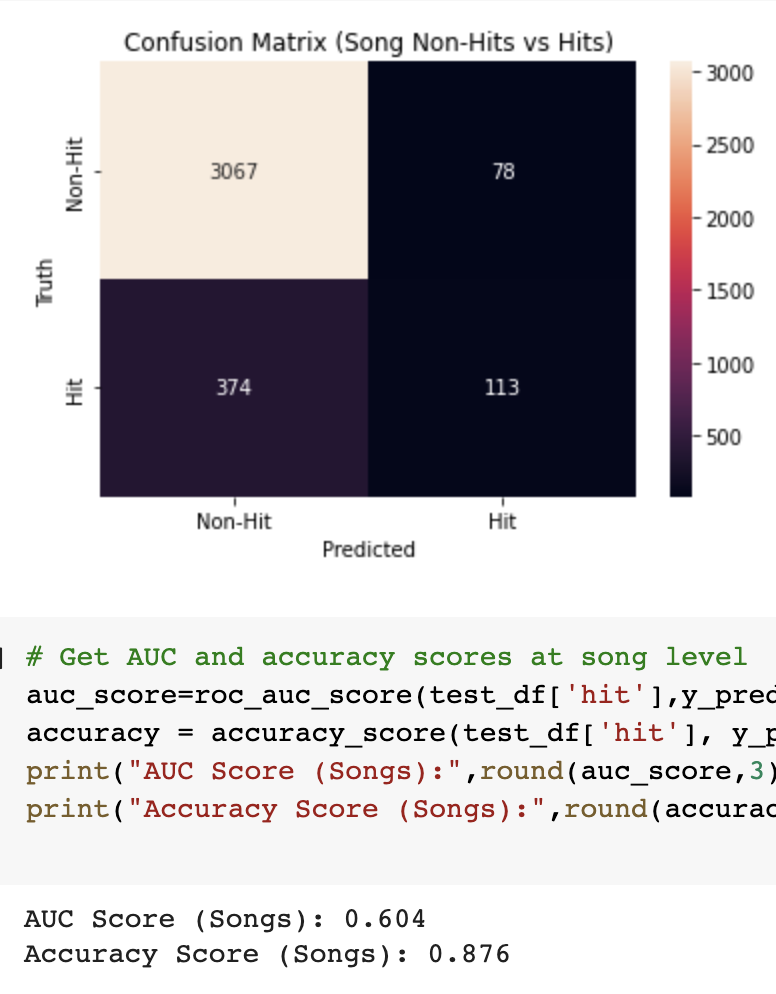
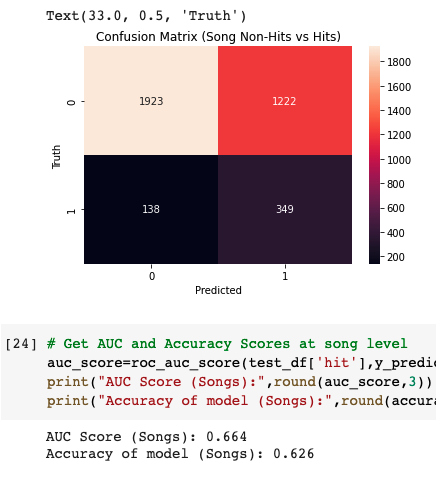


**Changing train layers**

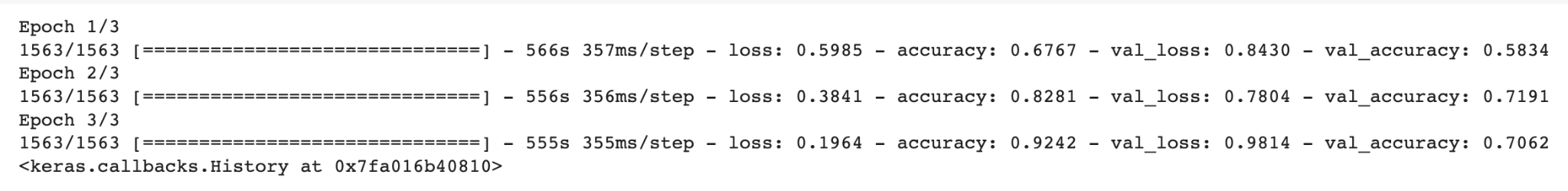
* New chunk/prob\_aggregation code – **Full dataset (RUN 3)**
  + **Train layers 1**, learning rate 5e-5, dropout 0.1



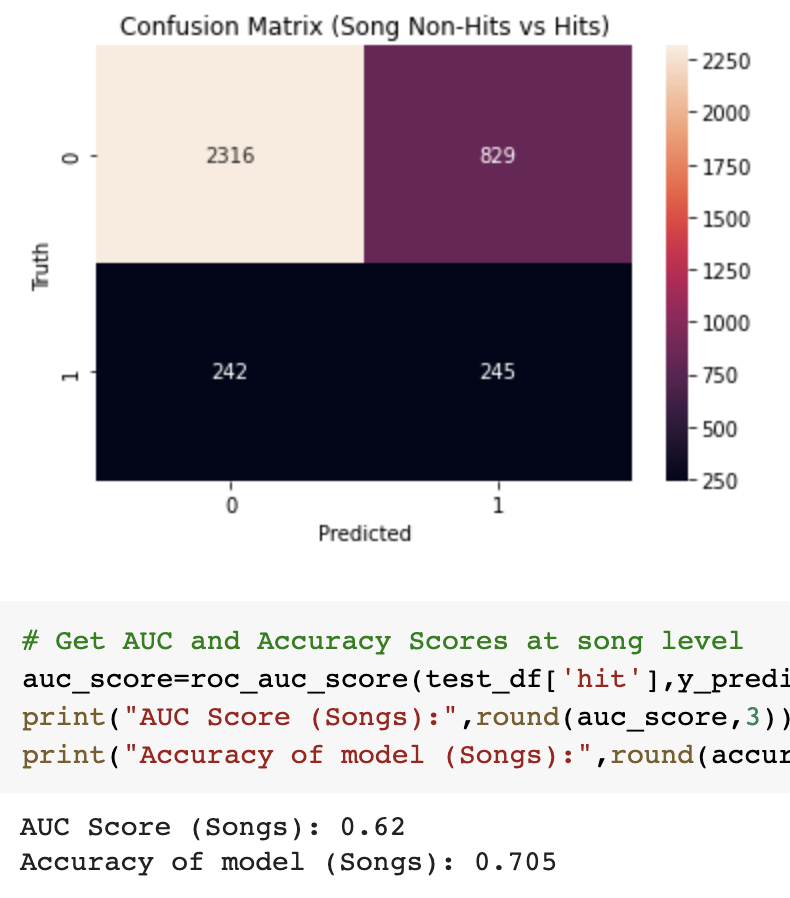
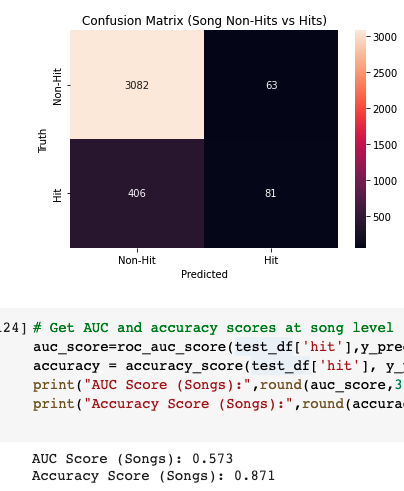
* Average vs regression



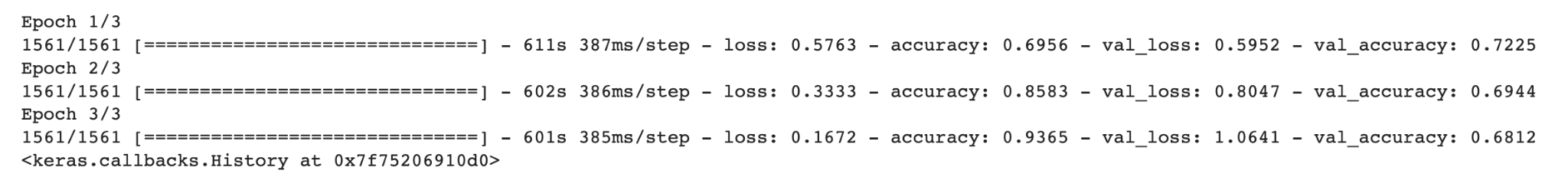
* New chunk/prob\_aggregation code – **Full dataset** 
  + **Train layers 2**, learning rate 5e-5, dropout 0.1

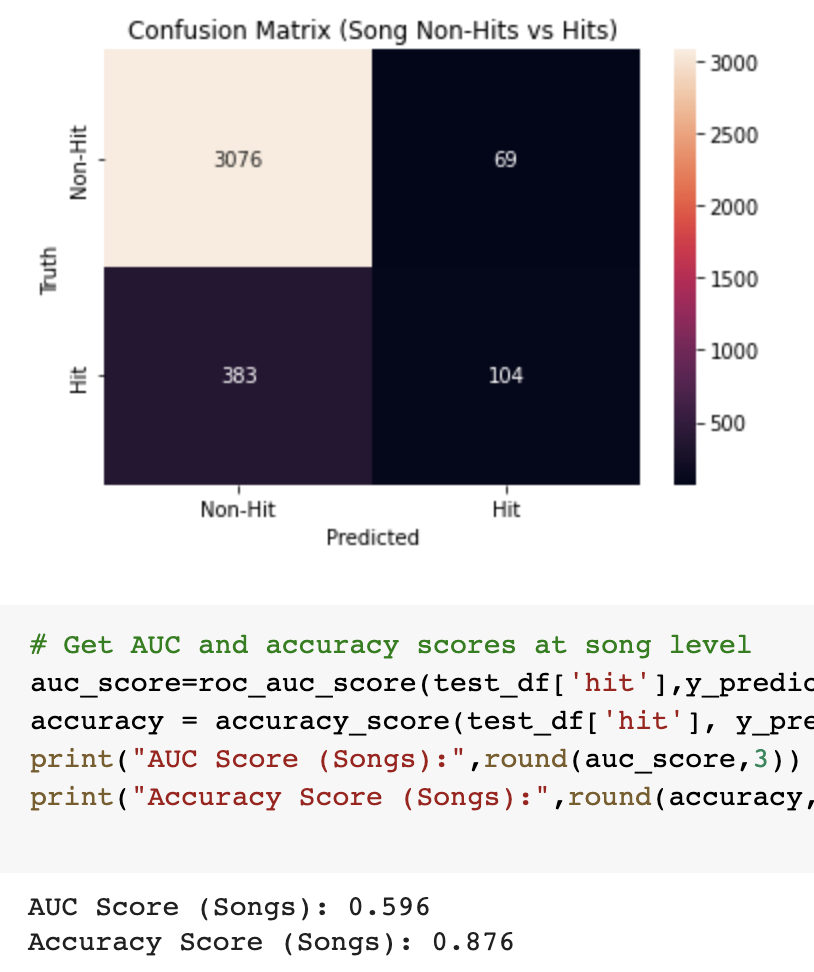
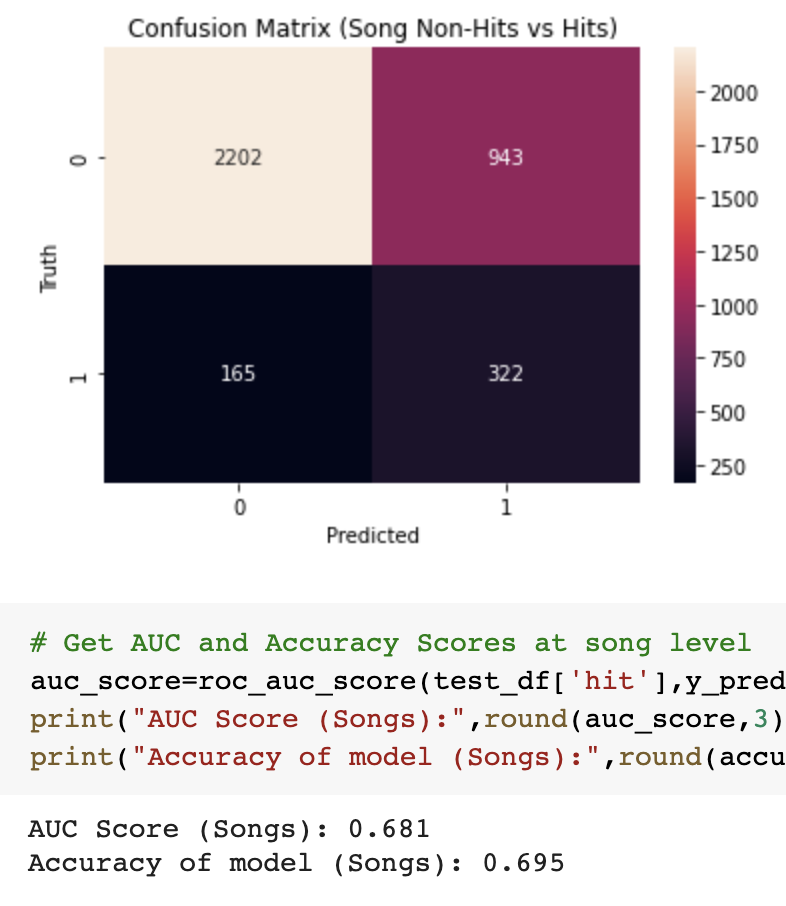


* Averaged vs Regression

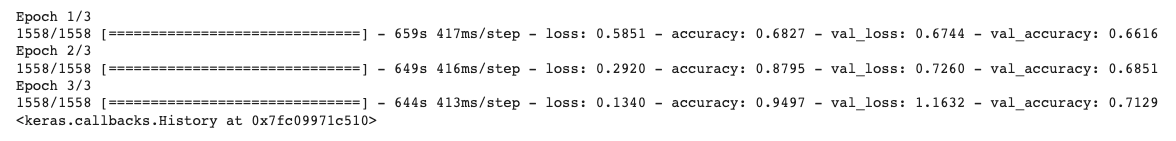
 

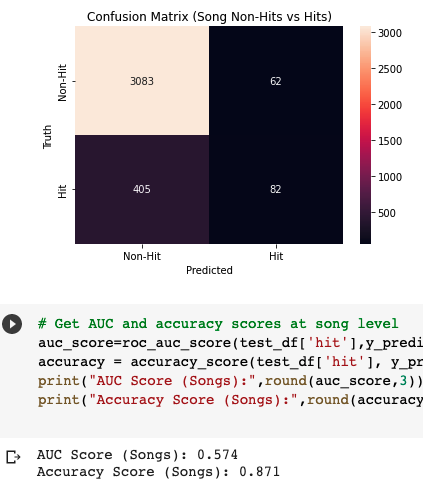
* New chunk/prob\_aggregation code – **Full dataset** 
  + **Train layers 3**, learning rate 5e-5, dropout 0.1





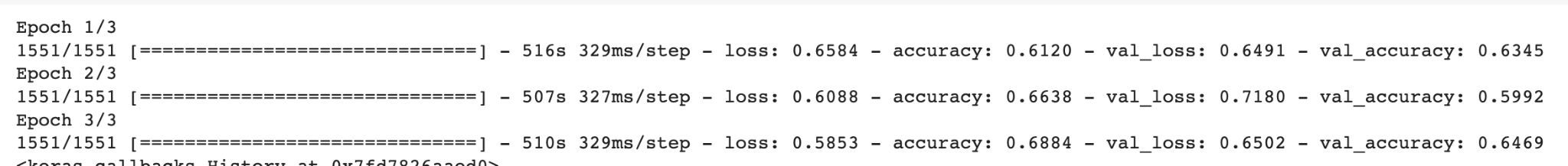
* New chunk/prob\_aggregation code – **Full dataset** 
  + **Train layers 4**, learning rate 5e-5, dropout 0.1

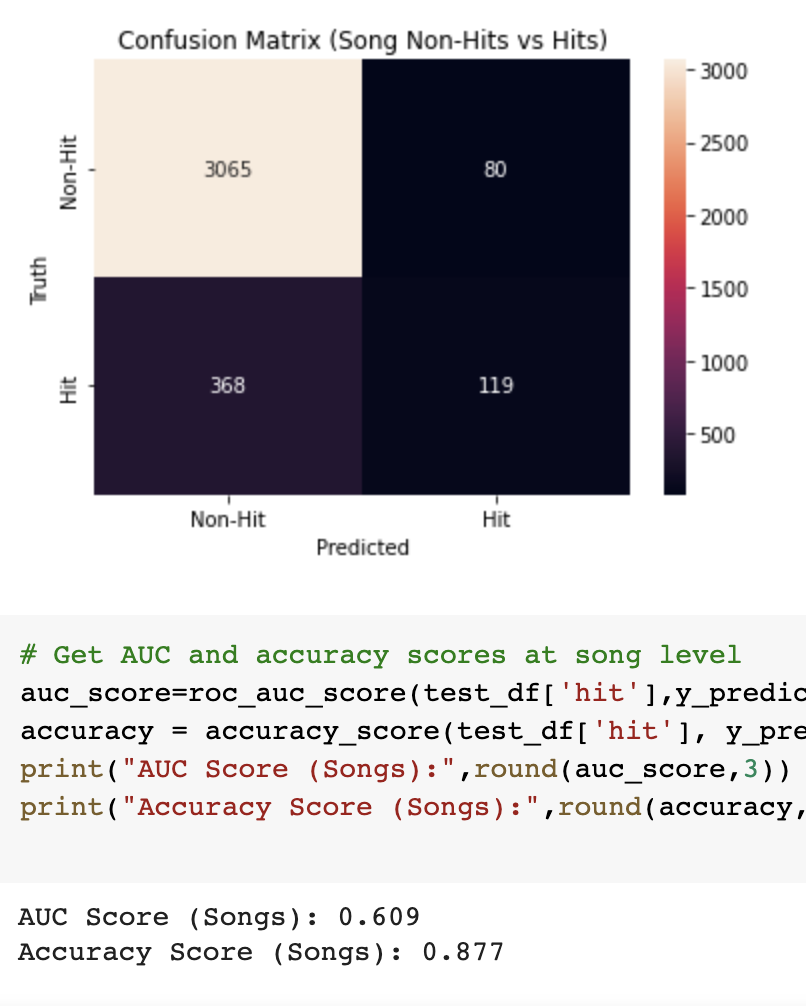




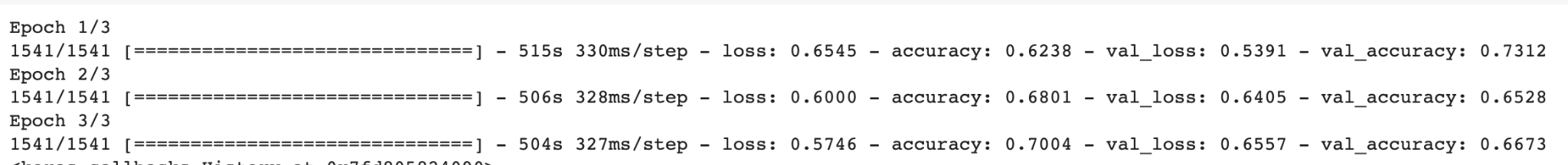
**Changing learning rate**

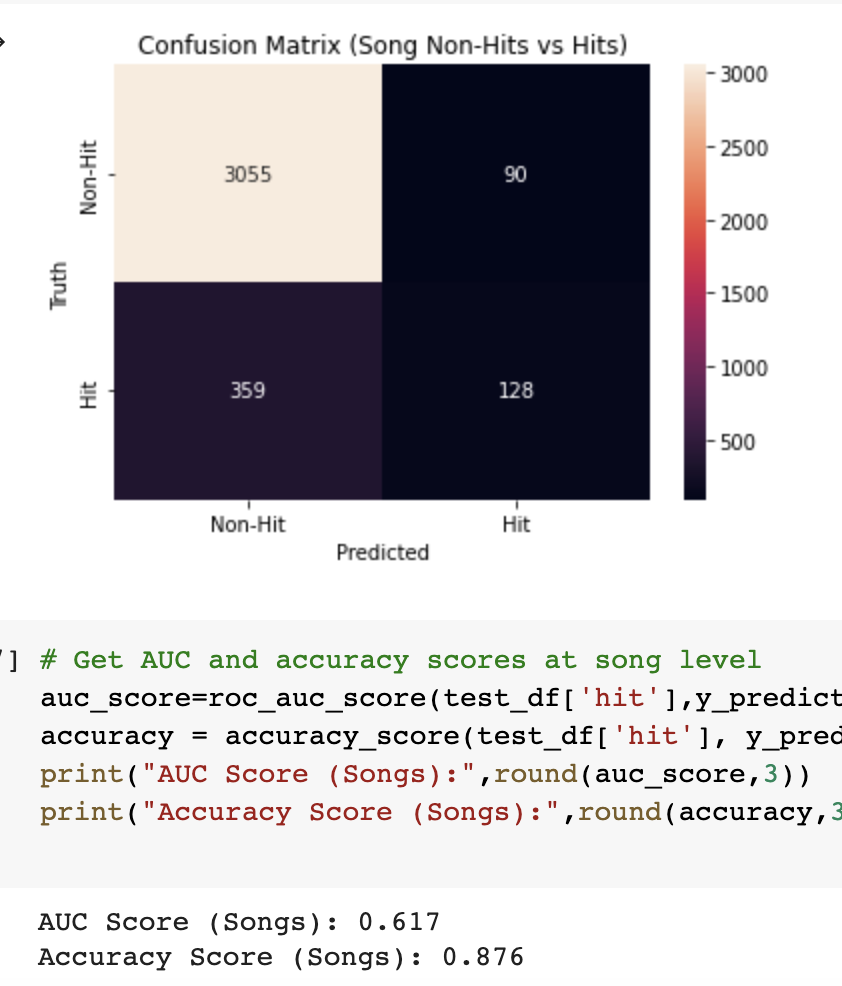
* New chunk/prob\_aggregation code – **Full dataset (ONLY REGRESSION)**
  + Train layers 1, **learning rate 5e-6**, dropout 0.1





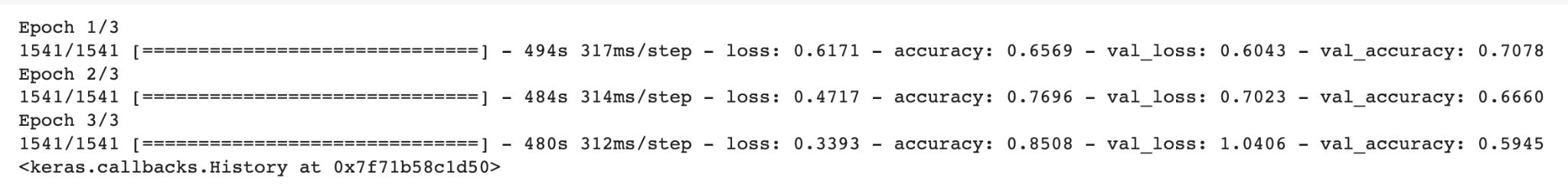
* New chunk/prob\_aggregation code – **Full dataset (ONLY REGRESSION)**
  + Train layers 1, **learning rate 5e-4**, dropout 0.1

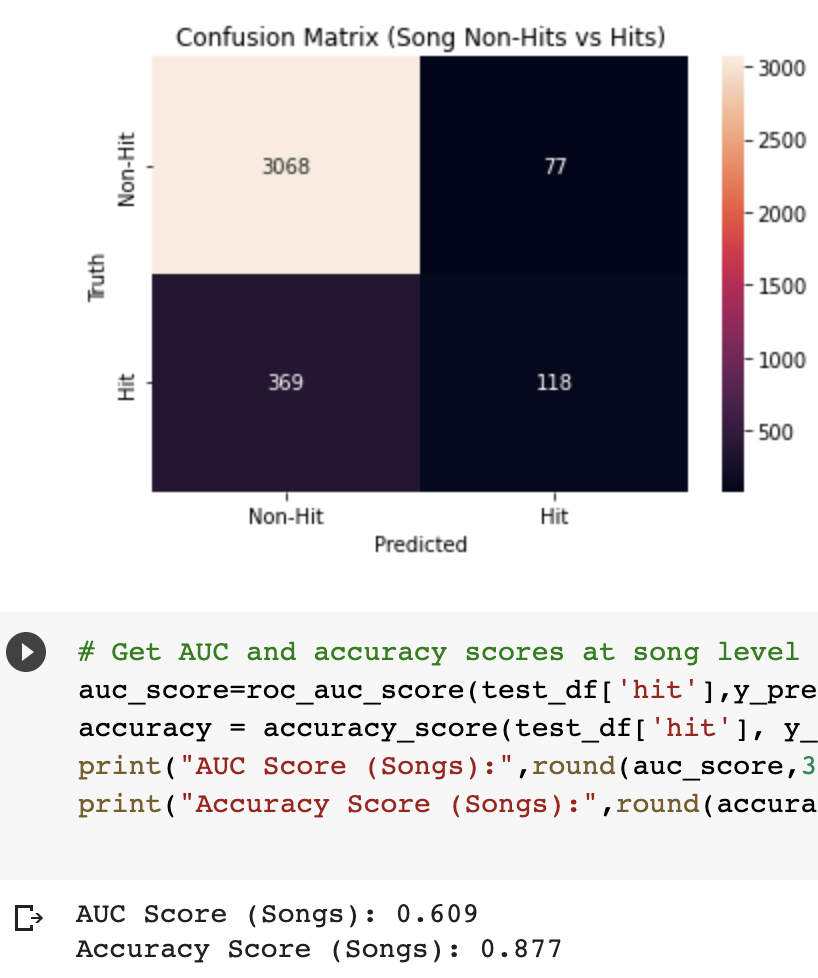




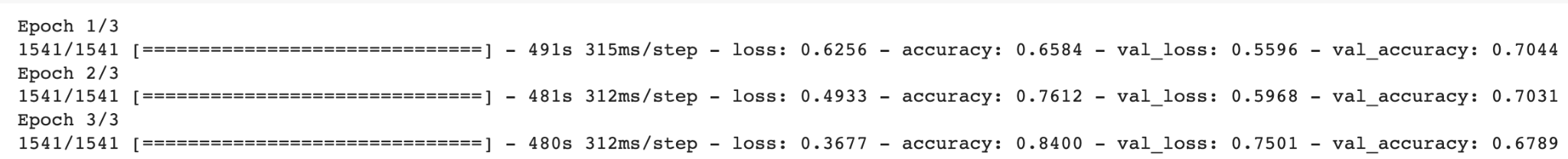
**Changing dropout rate**

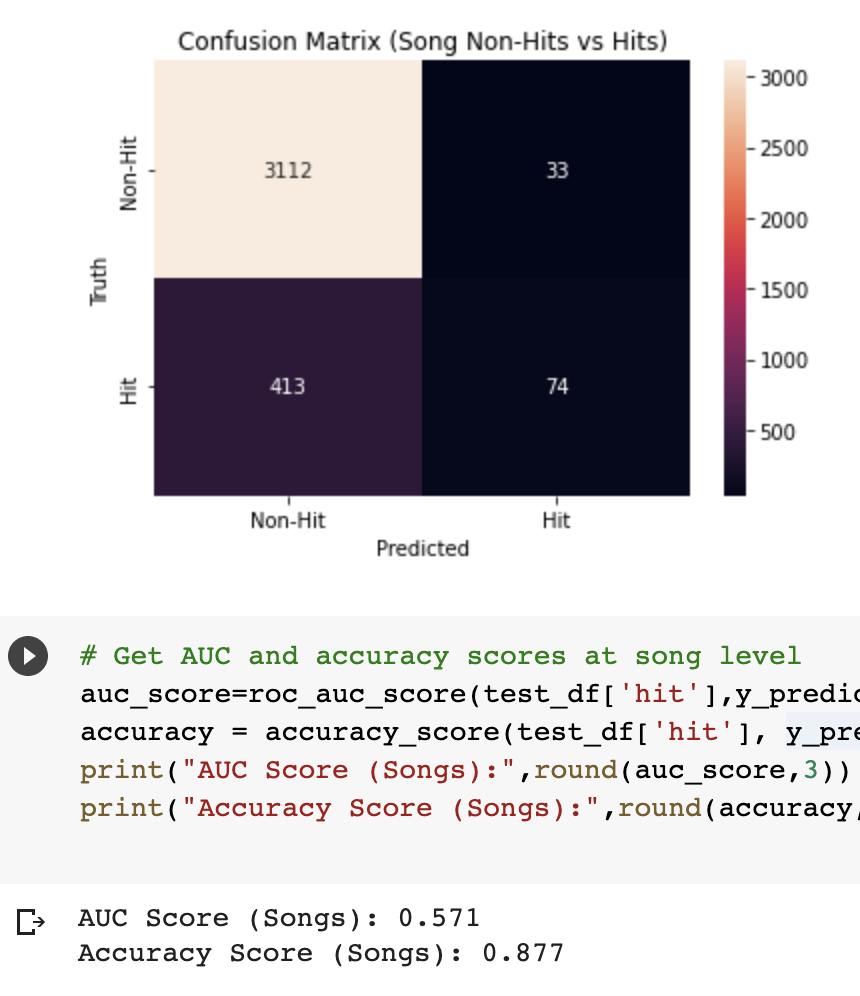
* New chunk/prob\_aggregation code – **Full dataset (ONLY REGRESSION)**
  + Train layers 1, learning rate 5e-5, **dropout 0.05**



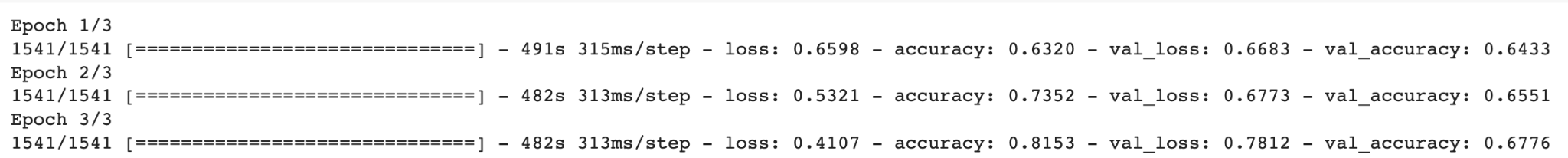


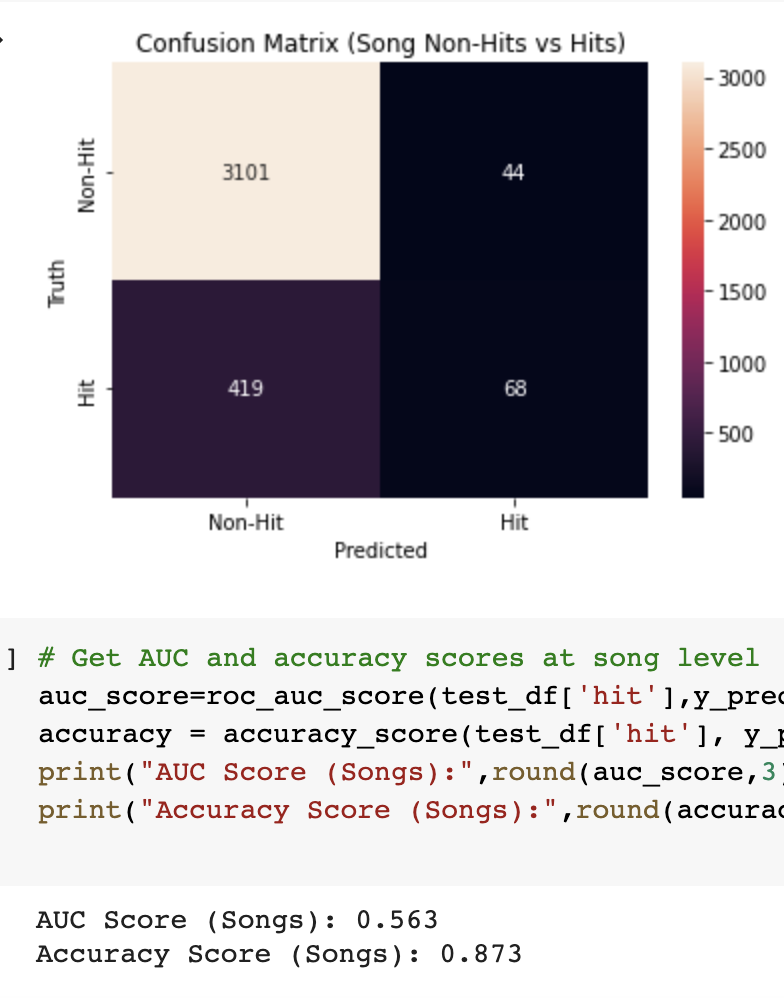
* New chunk/prob\_aggregation code – **Full dataset (ONLY REGRESSION)**
  + Train layers 1, learning rate 5e-5, **dropout 0.2**





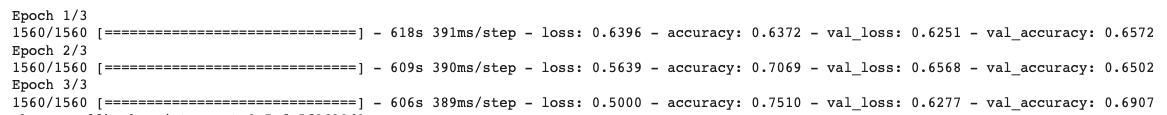
* New chunk/prob\_aggregation code – **Full dataset (ONLY REGRESSION)**
  + Train layers 1, learning rate 5e-5, **dropout 0.5**





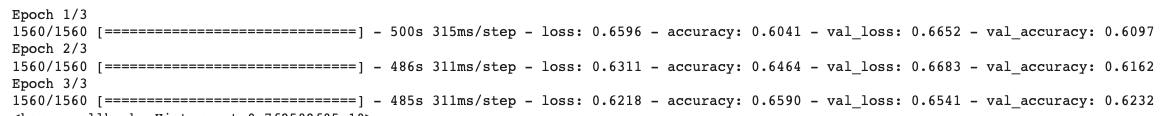
**FINAL FINE-TUNED**

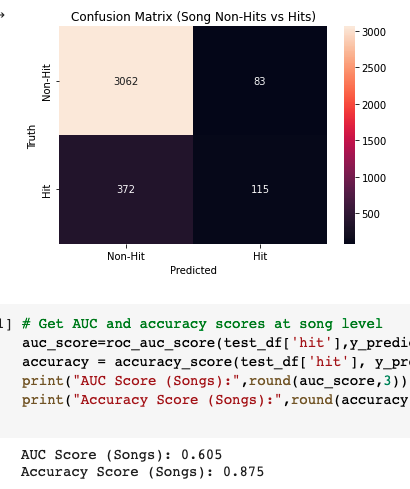
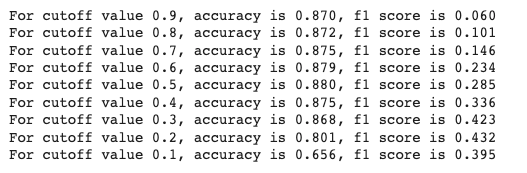
* train layers 3, learning rate 5e-6, dropout 0.2



**FINAL NO FINE-TUNING WITH METADATA**

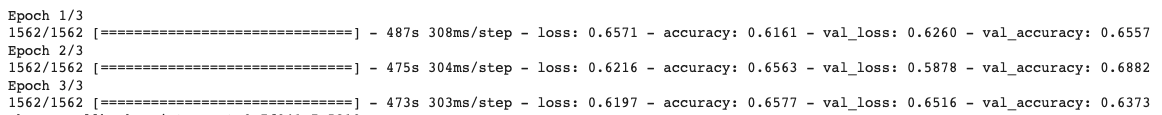
* train layers 0, learning rate 5e-5, dropout 0.1

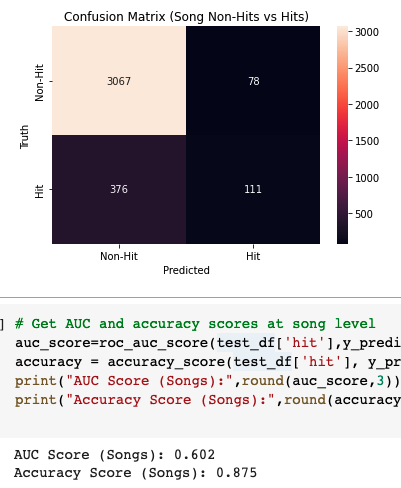
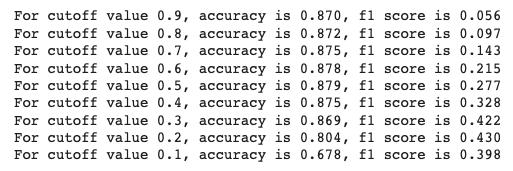


**FINAL NO FINE-TUNING WITHOUT METADATA**

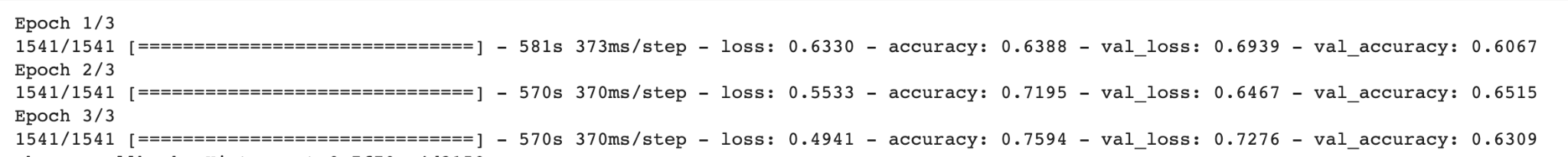
* train layers 0, learning rate 5e-5, dropout 0.1

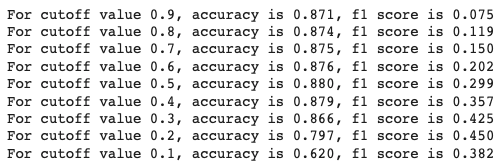
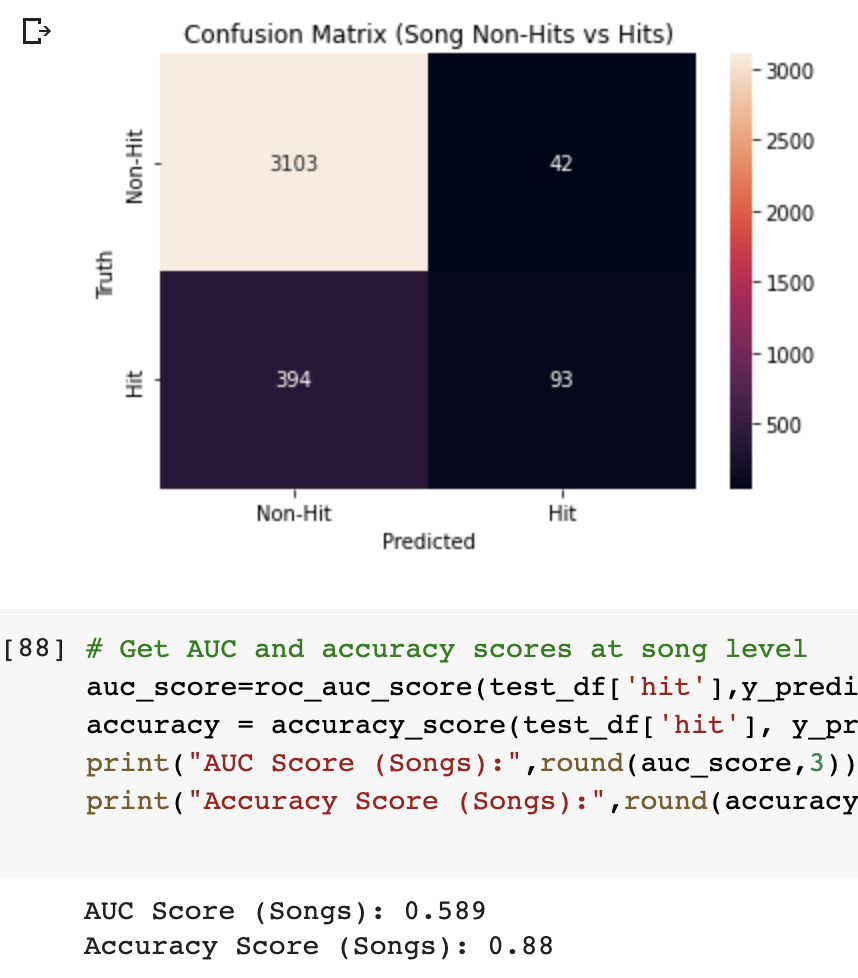




**FINAL FINE-TUNED WITH METADATA**

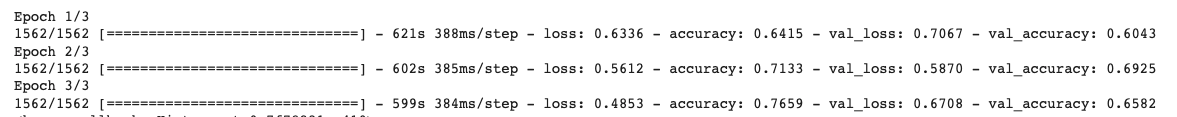
* train layers 3, learning rate 5e-6, dropout 0.1

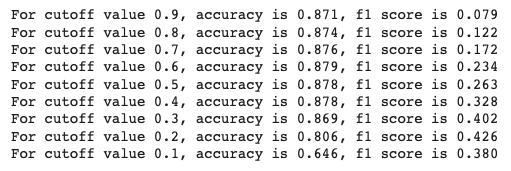
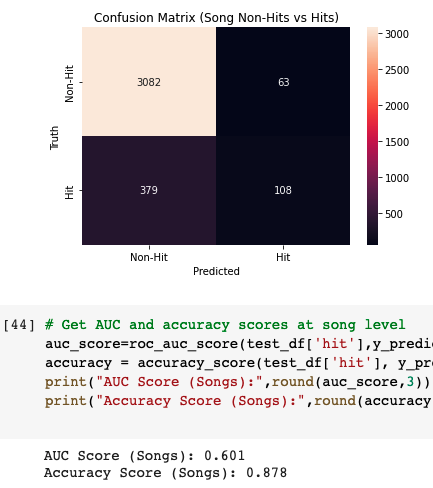




**FINAL FINE-TUNING WITHOUT METADATA**

* train layers 3, learning rate 5e-6, dropout 0.1





* Why we chose BERT for this task as opposed to SVM/Naive Bayes
  + BERT IS A BETTER FEATURE EXTRACTOR THAN TF-IDF because it’s been pre-trained so much on large corpus of data with a more complex loss function
  + **BERT can be used as a feature extractor to extract features from text to feed into a logistic regression model**
  + TF-IDF is very shallow features (looking at term frequencies in a document and only at numbers) whereas BERT has been pre-trained on filling in missing tokens and has a much better understanding of the English language
  + Take BERT model and train with a 2-layer neural network; they take TF-IDF features instead and train 2-layer neural network with the TF-IDF features
  + SVM is close to 2-layer neural network
  + we tried TF-IDF with a 2-layer model (SVM) and it didn’t perform well and now we’re looking for a better feature extractor
    - Replace naive bayes/logistic regression with SVM
    - from sklearn.svm import svc
    - **2-layer neural network is same as SVM with RBF kernel**
    - you tried SVM with TF-IDF features, didn’t perform as well, hypothesis was that we needed better features, and current best model for feature extraction from text was BERT - instead of TF-IDF with SVM, we tried BERT with a 2-layer neural network on top
    - **BERT allows us to extract robust features by applying a stack of self-attention layers to input text**
  + **Ask Mark about SVM - SVM strength in text analysis and closer to neural network**
  + **Feature importance from SVM**
* How the dropout, layers, and learning rate affected the model’s performance
  + Layers
    - you’re backpropagating and updating the lengths more
    - **The more layers I unfreeze it’s looking more at MY data as opposed to the BERT data**
  + Dropout
    - randomly dropping values to 0
    - too much regularization above 0.2 - will need to look this up
  + Learning rate
    - all of these hyperparameters need to be tuned
    - pre-trained BERT model has already learned transformation and you don’t want to train it with high learning rate because you’re ruining rates that it already has
      * catastrophic forgetting!
    - correlation between learning rate and frozen layers - have lower learning rates for BERT layers and higher learning rates for new layers, the more layers unfrozen, the lower learning rate needed otherwise loss will skyrocket
      * per-layer learning rate - dense layers is high learning rate and BERT layers is a low learning rate
* BERT Architecture
  + input IDs are translated into embeddings and huge embedding matrix is trained in addition to the model
  + input = batch size x sequence length (wordpiece tokens)
  + translates each input id into a random embedding
* Results Analysis
  + Show PR curve for Naive Bayes compared BERT model
  + Look at ROC AUC and average precision
* Picking the split - we choose partitioning method based on how we want to apply this to production; maybe split validation set up by later years as well

