Artificial Intelligence In Mood Prediction For Music Shuffle Optimization

Mood Association and Cognitive Response in Individuals to Auditory Stimulation

| Saransh Rakshak |
Psychology C120 | Cognitive Science C100
Basic Issues In Cognition
Professor Davina Chen, Ph.D.
Fall 2021

Table of Contents

Our attempt at developing an AI model, as well as an interactive version of this project write-up can be found at the following GitHub Repository: https://github.com/saranshrakshak/cog_sci_c100.git

I: Background & Biological Processes

Music is a part of our lives daily. Our goal is to develop a method- using artificial intelligence, data analytics, and training models- in which we are able to track individual listening habits, correlate moods associated with certain types or genres of music, and create an optimal playlist, filled with either songs or low volume ambient-type music, which corresponds to that individual's mood. In this manner, we hope to increase user listening time by increasing the individual's 'connectedness' with the music and enticing them to listen to more and more songs.

A) Auditory Stimulation

Our stimulus, sound, propagates through space by compressing and expanding air molecules, sending them outwards from the source in all directions in the form of a pressure wave. This pressure wave is perceived by the auditory system as a sound wave, and can be broken down into two functioning variables: loudness and pitch.

Volume can be determined from the amplitude of the stimuli sound wave, or how much change in Y occurs given a range X (wavelength), while pitch is derived to be the frequency of the sound wave, or the length of X to complete one cycle (or wavelength). **Pitch** is measured in Hertz (Hz), and limitations in the human auditory system allows us to perceive sounds in the 1000-5000 Hz range with relatively greater sensitivity.

B) Anatomy of the Ear

Past the *pinna*, we reach the middle ear, consisting of three small bones (*ossicles*), called the *malleus* (hammer), *incus* (anvil), and *stapes* (stirrup). The *ossicles* act as an amplifier to small mechanical vibrations of air molecules, and sensitivity of the *ossicles* can be changed by contracting muscles in the ear to filter out unwanted sounds (for instance, contracting when loud noise prevents damage).

Further down in the ear, we reach the *cochlea*, made up of a coil of three parallel canals,- all which branch to the *vestibulocochlear nerve (VIII)*. Sound vibrations of *the oval window* cause a wave to pass through the *vestibular canal*, and travel back down the *tympanic canal* to the *basilar membrane.* This is approximately 5 times wider at the apex than at the base (even though the *cochlea* grows narrower towards the apex), thus higher frequencies cause maximum displacement of fluid near the base, while lower frequencies cause the greatest displacement at the apex. The *cochlea* then works to translate this *temporal* signal into a *spatial* signal.

C) Mechanics of the Ear

Auditory neurons in all levels of the auditory system display *tonotopic organization*- they are spatially arranged in a corresponding map to the frequencies which they respond to. Both ears have two rows of sensory cells, a row of approximately 3500 *inner hair cells(IHC)*, and three rows of approximately 4000 *outer hair cells(OHC)*. Each *IHC* is associated with around 18 to 20 different sensory auditory nerve fibers, responsible for the perception of sound, as well as some motor fibers for the inhibition of some sound stimuli. Meanwhile, *OHC* have very few sensory fibers, rather, stimulation of *OHC* motor fibers cause *OHC*'s to change in length, thereby stiffening or relaxing the *basilar membrane* and fine tuning particular sound frequencies as needed.

II: Cognition & Auditory Processing

A) Psychological States (Moods)

An experiment in the late 1980's conducted by Valerie N. Stratton and Annette H. Zalanowski monitored the effects that pleasant, depressing and no music had on subjects while they were viewing a piece of art. Subjects, while viewing the art and listening to a certain type of background music (or no music), were then asked to create either a happy story, a sad story, or whatever came to the subjects mind. The experiment saw that when the subject was asked to create a neutral story (i.e. whatever came to mind), music would determine the mood change. So, if the person was asked to tell a neutral story while being surrounded by sad background music, the subject's story would be sad to reflect the room. However, the Stratton-Zalanowski experiment also saw that when the subjects were asked to create an expletively sad story or happy story- then the type of music that surrounded them did not alter the motif of the story told by the subject. As a result, the experiment proved that "mood responses to music are indeterminate and depend on cognitive processes"- an assumption that we will incorporate in our model. (Stratton 122)

B) Cognitive Response

Seibert and Ellis, in 1991, developed a technique for testing and conducting research on *mood-congruent cognition*- in other words, the relation that people conduct their behavior, interpret, acquire, and store information to memory. Additionally, the Seibert-Ellis experiment expanded on the Stratton-Zalanowski experiment, as they were able to show that the induction of happy or sad moods provided poorer recall(memory) than a neutral mood.

Table 8.2
Synopsis of MCI studies

			Initial Cognitive Task				
Study Number	*Chief Concern*	*Sample Size*	*Session 1*	*Session 2*	*Mood Measures*	*Genuineness Ratings*	*Thought Accounts*
1	MC	36	CEG	CEG	AG	no	no
2	MC	60	UEG	UEG	AG, PANAS	yes	yes
3	MD	48	G/R	FR	AG	no	no
4	MD	64	WR	SC	AG	no	no
5	MC, MD	64	UEG	FR	AG	no	no
6	MC, MD	128	UEG	FR	AG	no	yes
7	MC, MD	72	UEG	CP	AG	no	no
8	MC, MD	64	UEG	CP	AG, PANAS	yes	yes

Note: MC = mood congruence; MD = mood dependence. Sample size = total number of participants (all experimental conditions combined). CEG (UEG) = constrained (unconstrained) autobiographical event generation; G/R = generate/read; WR = word rating; FR = free recall; SC = stem completion; CP =

Table 8.3
Selections of happy and sad music used in the MCI

Music	*Set*	*Composer*	*Composition*	*Duration (min:sec)*
Happy	1	Mozart	*Eine Kleine Nachtmusik*: Allegro	5:38
	1	Mozart	*Eine Kleine Nachtmusik*: Rondo	3:01
	1	Mozart	Serenade #9: Finale	4:26
	1	Bach	*Brandenburg Concerto* #3: Allegro	4:57
	1	Tchaikovsky	*The Nutcracker*: Waltz of the Flowers	6:36
	1	Tchaikovsky	*The Nutcracker*: Trepak	1:00
Happy	2	Beethoven	Symphony #9: Presto	2:36
	2	Vivaldi	*Four Seasons*: Spring I Allegro	3:01
	2	Vivaldi	*Four Seasons*: Spring III Allegro	3:56
	2	Vivaldi	*Four Seasons*: Autumn I Allegro	4:24
	2	Vivaldi	*Four Seasons*: Autumn III Allegro	3:10
	2	Tchaikovsky	*The Nutcracker*: Dance of the Flutes	2:04
	2	Tchaikovsky	*1812 Overture* (excerpt)	2:04
Sad	1	Albinoni	Adagio in G Minor	4:52
	1	Sibelius	Violin Concerto: Adagio di Molto	3:14
	1	Lehar	Vilja-Lied	2:23
	1	Schumman	Traummeri	3:31
	1	Dvorak	Symphony #9: Largo	5:50
Sad	2	Grieg	*Peer Gynt*: The Death of Ase	5:49
	2	Tchaikovsky	*Swan Lake*: Dances des Cynges	3:00
	2	Chopin	Prelude #4 in E Minor	2:00
	2	Vivaldi	*Four Seasons*: Autumn Adagio	2:20
	2	Stravinsky	*Firebird*: Lullaby (excerpt)	3:15

Additionally, we will be referencing data established in the development of the MCI Technique (also known as the *continuous-music technique*), a subsection of tracks used in the original MCI Technique is shown in Table 8.3. (Coan 129)

Table 8.4
Minutes to achieve mood criterion as a function of mood condition, participant gender, and study segment (studies 1–8 combined)

Mood Condition	*Participant Gender*	*nS1/nS2*	*Study Segment*	
			Session 1	*Session 2*
P	Women	185/181	15.8 (9.3)	14.5 (7.5)
P	Men	83/87	15.2 (8.1)	15.7 (9.5)
U	Women	181/185	16.7 (8.8)	17.4 (7.7)
U	Men	87/83	18.6 (8.7)	18.3 (8.3)

Note: P = pleasant mood, U = unpleasant mood. nS1/nS2 = number of participants per mean value in session 1 (S1) and in session 2 (S2). Standard deviations are enclosed in parentheses.

While the original purpose of MCI was to alter mood by a certain song, we will be using its reverse: seeing the qualities of a certain song determined by the subject's behavior and listening pattern, and subsequently determining the subject's mood. Additionally, we will be seeing data collected from previous MCI Technique studies to ensure that the user continues listening to music for as long as possible - i.e. the user can still function in an optimal state in congruence with music, else if the music is too hindering to their current psychological state, they would likely stop the music from continuing playing. Results from these previous MCI studies are shown in Table 8.2. (Coan 128) Results for various prior MCI studies are summarized in Table 8.4. (Coan 130)

Note: Difference in Men/Woman sample size may decrease confidence level and/or increase margin of error. **In our case, we assume this to be negligible.**

C) Data Collection

The data set and visuals can be viewed in its entirety via the Jupyter Notebook (stored in the GitHub Repository linked above).

Our data heap was composed of the following data points; each column corresponds to a different type of data point (i.e. type of music, duration of music, user or random selected, common correlating mood associated with the type of music, etc.), and each row is an instance/subsection of that individual's listening cycle for an individual iteration of a song (as song can be played multiple times if criteria met in each instance, however cannot be repeated - our current song cannot equal the next song, but possibly can be somewhere else in Queue).

There will be no initial survey-type input of the user' mood, as that would defeat the purpose of the learning model and would not allow for an adaptive shuffle algorithm based on predicted mood. Rather, the algorithm will treat each run of the deck of music as an instance of a global queue. Initial user input will be read by our algorithm for one of two possibilities: ***Active User Song Selection,*** or ***Initial User Shuffle Launch***.

III. Learning Algorithm Overview

A) Module 3: Song

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**Module 3: Song**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

I. ***Active User Song Selection***: Upon running the program and accessing the deck of songs (**deck_songs**), our user has chosen a specific song to play.
   A. This song is our initial data point, as it is most likely representative of the users mood. The song is first analyzed for lyrics.
      1. **If** lyrics are present, run Sentiment Analysis on the lyrics with our *Natural Language Processing Tools Kit* API, using our *Lyric Analysis* file (**lyric_analysis.py**).
         a) Convert file into .wav format, using **convert_to_wav**(**filename**):
         b) Gather sentiment score for overall song, using **song.nltk_lyrics**.
         c) Pass to `Module 2: User`.
      2. **Else**, establish song mood with our predetermined database or analyzing Beats Per Minute of the song, using the assumption that songs with a higher beat count per minute are faster paced and cause more excitatory effects, while lower beat counts would signify a slower tempo song that is most likely composed of a depressing theme, resulting in more inhibitory effects.
         a) **If** this song is in existence in our song database, it gains a higher song mood certainty score as it is a pre-established title.
         b) Due to this assumption, method 2 has a lower resulting certainty score and thus a greater margin of error.
            (1) **If** this song reaches a certain margin of error, drop the song mood entirely.
            (2) **If** this song gains a high enough certainty, add song and mood key-pair into our song database.
               (a) **If** song certainty passes validity check **threshold** (based on user certainty, user mood, and song certainty) for user mood, add this song into our global song_mood_db.
                  (i) This can only be done by passing our song into our `1st module` initially, then passing this song over from `module 1` to our global database (song_mood_db) of song-mood key-pairs.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

(a) Same ***Admin*** (including */master*) cannot do both these processes, i.e. they can either pass this song from `module 3` to `module 1: Queue`, or from `Module 1: User` to global.

(i) Memory safety vulnerability mitigation.

c) Pass values to `Module 2: User`.

II. ***Initial User Shuffle Launch***: Upon running the program and accessing the deck of songs (**deck_songs**), our user has chosen to shuffle the entire deck of songs.

A. In this case, we will have no initial data point and automatically pass our process up to `Module 2: User`.

B) Module 2: User

~~~~~~~~~~~~~~~~~~~~~~~~~~ ***`Module 2: User`*** ~~~~~~~~~~~~~~~~~~~~~~~~~~

`Module 3: Song` has passed a song into User, with its corresponding certainty score and song-mood allocated key-pair. Our song has been given a score, based on our song's mood value, and our certainty of our song's mood value. `Module 2: User` will develop values for our user's mood and user's mood certainty.

I. **If** this user is in existence in our global database, we will use the most recent user queue cached in our history, and bypass `Module 1 Playlist` for our first iteration (run) of a song. The first song from the old playlist (saved in **queue_songs**) will be treated as a chosen song by the user, and the algorithm will backpropagate and restart for the next song in `Module 3`.

II. **Else**, we will assign the user as a new instance, and determine his mood based on the occurrence of the first song which is played more than 70% through.

A. **If** the user listens to a certain type(mood) of a song, the user will get a greater value for the predicted user-mood which matches this song.

1. Additionally, if more plays of songs with , our user-certainty value will increase, and be passed into `Module 1: Playlist`.

B. **Else,** if the user does not show any consistent pattern , we will bypass `Module 1` and pass a random, shuffled playlist into our global **queue_songs** uand rerun our modules until we find an optimal playlist for queue songs. This cannot occur until a pattern can be established with more instances of new songs and greater frequencies of similarly predicted mood songs.
~~~~~~~~~~~~~~~~~~~~~~~~~~

C) Module 1: Playlist

~~~~~~~~~~~~~~~~~~~~~~~~~**Module 1: Playlist**~~~~~~~~~~~~~~~~~~~~~~~~

Module 2: User has passed in a predicted user mood, and associated user certainty for all individual users. Module 1: Playlist will run comparisons in order to generate an optimized playlist to pass into our global **queue_songs**.

1. **If** our predicted songs mood equals our predicted user mood AND
    a. **If** our user-mood certainty is greater than the preassigned user-mood threshold (stored in user.mood_thresh) AND
        i. **If** our song-mood certainty is greater than the preassigned song-mood threshold.
            1. Add mood-song key-pair to our cached, global directory **song_mood_db**
            2. Pass song into our main playlist for this user, global **queue_songs**
                a. This is a global variable, which gains values from occurrences of modules by an individual
                    i. Thus, it can save a localized directory for each independent user's **song_mood_db**.
                b. Global value outside of modules ensures preservation of both our directory **song_mood_db** and global playlist **queue_songs** for each independent user.
2. **Else**, rerun modules with null values and start fresh for the user -assigning new values to songs, users, predictions and certainties- and thus creating a more accurate song-user key-pairs for the same session.
    a. See Machine Learning- Error Correcting Code *pg.17* for details on how we are handling changes in an individual user's mood within the same session.
    b. Allows for optimization of playlist when users mood changes in the same listening session.
~~~~~~~~~~~~~~~~~~~~~~~~~

IV: Constructing The Neural Network

A) Converting Raw Data

To see visuals corresponding to data conversion and analysis, please refer to the Jupyter Notebook linked above.

Our AI model will correlate mood associated with input sound determined by a series of following embedded `modules`:

global

~~~~~~~~~~~~~~~Module 1: Queue~~~~~~~~~~~~~~~

`Module` with a collection of valid songs that match the user's cognitive state, queued with the most significant and highest certainty song at the head of the playlist stack, in location song.next_song and queue_songs[1].

~~~~~~~~~~~~~~~Module 2: User~~~~~~~~~~~~~~~

Use of adapted machine learning algorithms to analyze if the current song matches user preferences, or if the song is skipped or user selected. If the song satisfies conditions by meeting thresholds determined by our weights and biases(i.e. user.valid_mood == TRUE), our `module` pushes the song to our **`Queue Module (1)`** as a valid song to be played, else the song is dumped (skipped).

~~~~~~~~~~~~~~~Module 3: Song~~~~~~~~~~~~~~~

Our deepest embedded `module`, used in the analysis of the current song user is listening to.

B) Variables & Functions

Note: Helper functions, such as get_song(), get_user(), and get_volume() have been omitted.

~~~~~~~~~~~~~~~*Global Variables and Functions*~~~~~~~~~~~~~~~

★ (global) **song :** Our current playing song.

★ (global) **queue_songs :** Our queued playlist that we will be building using our AI model. Value at the 0 index is our current song, and value at index 1 is our next song to play.

★ (global) **deck_songs :** Unqueued deck of all available songs that we can play.

★ (global) **user :** The current individual user listening to music.

★ (global) **song_mood_db :** Key-pairs of song-mood correlations preserved with attached certainty scores, computed as an average between all runs of our program/learning algorithm.

~~~~~~~~~~~~~~~Module 3: Variables and Functions~~~~~~~~~~~~~~~

➢ song**.cur :** The current song being played.

➢ song**.nxt_sng :** The next song in our queue.

➢ song**.is_usr_selected :** A boolean, True if current song is user selected/user cued

➢ song**.is_skipped :** A boolean, True if user decides to skip current song, False if user listens to more than 70% of user.avg_listen_time of the song. A True (skip) would mean the user does not like the current playing song.

➢ song**.len_sng :** The length, in seconds, of the cur song.

➢ song**.type_sng :** The predicted genre for the current song.

➢ song**.bpm_sng :** The song's beats per minute, or pace. We are assuming that songs with a lower bpm_sng value will be more likely sad, while a high bpm_sng value will be more likely to be happy.

➢ song **.volume :** The volume of the current song and user audio.

→ ***Lyric Analysis***

➢ song**.nltk_lyrics :** Use of natural language tools kit API to analyze lyrics of song.cur, only if the song contains lyrics, gives a value = [happy OR sad OR neutral OR Null if song.cur does not contain any lyrics].

→ ***Threshold Check***

➢ song**.mood_thresh :** The threshold needed to be reached to associate a song with a certain mood.

➢ song**.pred_mood() :** Use of historical data and machine learning to determine moods commonly associated with type of cur song. Function returns a predicted mood for associated current song.

➢ song**.cert_mood() :** Function that returns our certainty (0<=C<=1) that the associated mood for cur song is correct.

➢ song**.valid_mood() :** Returns a boolean True if song.mood_thresh reached, else False.

➢ song**.edit_thresh(int** bias**) :** Overwrites the value of our song's mood threshold, determined by int bias, by adding specified bias to our original

threshold. Stores the value in song.mood_thresh. Returns a new song threshold value of type int.

~~~~~~~~~~~~~~~Module 2: Variables and Functions~~~~~~~~~~~~~~~

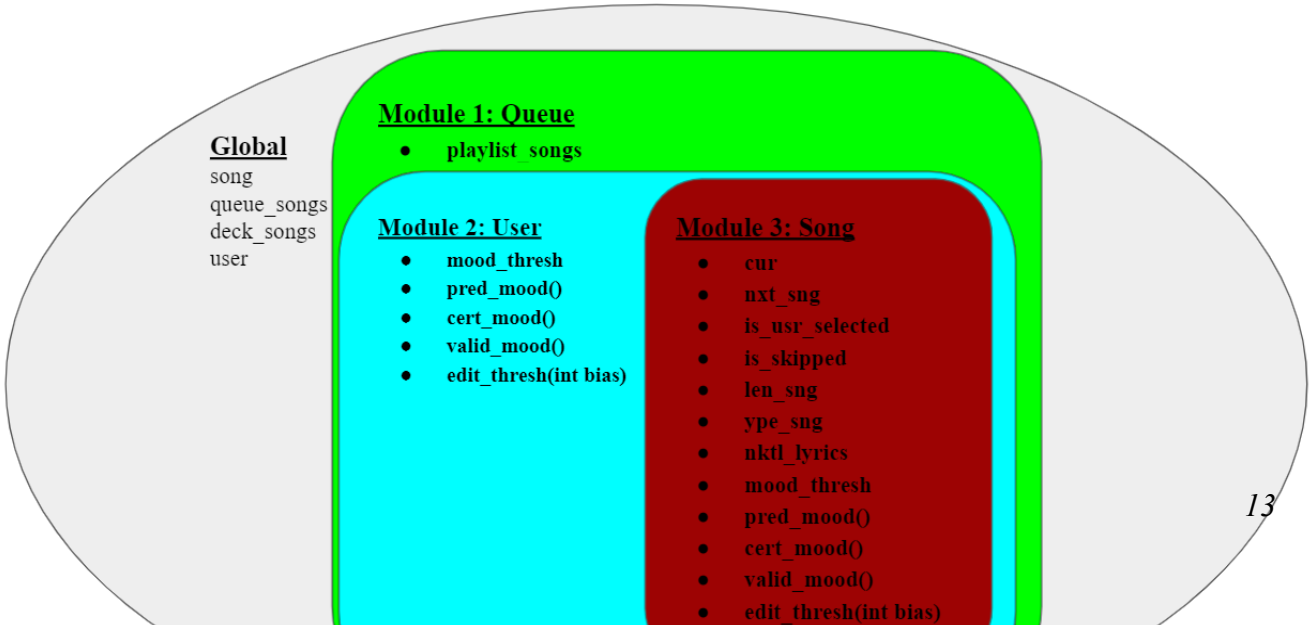
- user**.mood_thresh :** The threshold needed to be reached by the `module` to allocate the user to a certain mood.
- user**.avg_listen_time() :** The average listening time for all songs for our user.
- user**.pred_mood() :** Our prediction of the cur user's mood, determined by previous played song *asserting* song.is_usr_selected == True (i.e. we want to learn the users preference of songs to determine their mood, so we will only use songs the user actively selected as they are most representative of the user's internal mood and will give negative scores to songs/moods associated with songs where song.is_skipped == True).
- user**.cert_mood() :** Returns our certainty (0<=C<=1), as a double, that the user's predicted mood is correct.
- user**.valid_mood() :** Returns a boolean True if user.mood_thresh reached, else False (i.e. we are uncertain of users mood).
- user**.edit_thresh(int** bias**) :** Overwrites the value of our user's mood threshold by adding specified bias to our original threshold. Stores the value in user.mood_thresh. Returns a new int threshold value.

~~~~~~~~~~~~~~~Module 1: Variables and Functions~~~~~~~~~~~~~~~

- song**.playlist_songs :** Our queued deck of all available songs where
  - song.pred_mood() == user.pred_mood()
    AND
  - user.cert_mood() >= user.mood_thresh
    AND
  - song.cert_mood() >= song.mood_thresh

  Resulting playlist preserved in global **queue_songs**.

C) Variable Allocation Visualization



D) Module Access & Edge Weights

For detailed description of individual edge weight allocation, please refer to the Jupyter Notebook linked above.

~~~~~~~~~~~~~~~ Module 3 ↔ Module 2 ~~~~~~~~~~~~~~~

Note: Secure nodal interaction.

/master can edit both `modules` but cannot edit global instance variables. Assigned ***admin(s)*** can edit respective `modules`, but ***admin*** cannot edit global variables corresponding to unauthorized `modules`. For instance, `Module 2` access cannot give permission to edit `Module 3` global instance variables nor run its functions.

- Convert songs with lyrics to .wav
- Generate lyrics from .wav file with Natural Language Tools Kit (nltk)'s google audio to text, save as plain text file.
- Produce sentiment score for corresponding song using analyze_text(text_file) and SentimentIntensityAnalyzer().polarity_score(text_file) (provided by nltk)

~~~~~~~~~~~~~~~~ Module 2 ↔ Module 1 ~~~~~~~~~~~~~~~~

Note: Secure nodal interaction.

/master can edit both `modules` but cannot edit global instance variables. Assigned ***admin(s)*** can edit respective `modules`, but ***admin*** cannot edit global variables corresponding to unauthorized `modules`. For instance, `Module 2` access cannot give permission to edit `Module 1` global instance variables nor run its functions.

~~~~~~~~~~~~~~~ Module 1 ↔ Global ~~~~~~~~~~~~~~~

Note: Secure nodal interaction.

/master alone cannot edit global, and any individual ***admin*** only has access to edit a subsection of global variables that pertains to its corresponding `module`.

- song → playlist_songs
  - *Assert* song.pred_mood() == user.pred_mood()
    - Ensure that the song's predicted mood is the same as our predicted mood for the user.

- *Assert* **user**.cert_mood() >= **user**.mood_thresh
  - Verify that we have a reasonable assumption for our user's mood.
- *Assert* **song**.cert_mood() >= **song**.mood_thresh
  - Verify that we have a reasonable assumption for our song's mood.
  - Calculate any bias errors and incorporate them into the song threshold with our function **song**.edit_thresh().
    - queue_songs → playlist_songs
- Calculate any bias errors and incorporate them into the user threshold with our function **user**.edit_thresh().

- Calculate any bias errors and incorporate them into the song threshold with our function **song**.edit_thresh().
- Incorporating new bias, populate queue_songs with **song**.playlist_songs
  - deck_songs → playlist_songs
    - Unqueued deck of all songs.
      - Convert songs with lyrics to .wav
  - Generate lyrics from .wav file with *Natural Language Tools Kit (nltk)'s* google audio to text, save as plain text file.
    - Produce sentiment score for corresponding song using analyze_text(text_file) and SentimentIntensityAnalyzer().polarity_score(text_file) *(provided by nltk)*
  - user → playlist_songs
- Generate a finalized song list in **song.playlist_songs** and save into global **queue_songs.**

E) Activation Threshold & Edge Weights

To see a detailed description of individual `module` *layer activation threshold, and associated decision trie (tree) with incorporated bias, please refer to the Python files linked above.*

~~~~~~~~~~~~Module 3 Song: Threshold and Edge Weight~~~~~~~~~~~

Our initial `module` will scan our deck of songs and will make one of two decisions based on our song's certainty and preassigned threshold, stored in song.mood_thresh. As our learning model gathers more data, this threshold value will be altered using song.edit_thresh, allowing for a lower threshold for songs that match user mood, while increasing the threshold for songs that do not match the users mood.

~~~~~~~~~~~~Module 2 User: Threshold and Edge Weight~~~~~~~~~~~

Our second `module` will scan our deck of songs and will make one of two decisions based on our song's certainty and preassigned threshold, stored in user.mood_thresh. As our learning

model gathers more data, this threshold value will be altered using user.edit_thresh. If our user shows rapid changes in mood, for instance, the threshold will be raised so that fewer songs are added to the optimized queue (as the model is still unsure of our user's actual mood). If our user shows a stagnant mood, then the threshold will be lowered, as we are more sure of the users actual mood (thus have a greater certainty), and can allow for a greater amount of songs that match our users predicted mood to be added to our playlist.

~~~~~~~~~~~~Module 1 Queue: Threshold and Edge Weight~~~~~~~~~~

Our final `module` will be in charge of calculating bias to each individual user's instances. These bias values will be used for addition into and modification of mood_thresh for both song and user.

F) Security

The following security measures were taken to mitigate vulnerabilities of the AI system from attackers.

1. Use of a memory safe language: *python v3.10.0*.
2. Separation of Responsibility
    a. Individual ***Admin*(*s*)** only have access to subsections of `module` collection (aside from */master*).
    b. No one ***admin*** alone, aside from */master*, can edit all three `modules`, but ***admin*** alone can read (only) all `modules`.
        i. Can only edit authorized subsections.
    c. */master* does not have access to run program or change global instances without a second ***admin*** authorization.
    d. No one ***admin*** can edit or view all global instances or edit and run all three `modules`.
    e. */master* solely cannot edit non-`module` subsections (any of global), but */master* can edit and read all three `modules` by itself.
3. Stack Canaries and ASLR
    a. Native in *nltk* and *numpy* libraries. Enabled by default by our AI program.

V: Machine Learning

A) Threshold Modification

In order to ensure maximum listening time, our model/queue must adapt to changes in the users preferences. This can be done by lowering or raising thresholds required to send a song from our `3rd, song-checking Module` up to our `2nd, user-checking Module`; and also by changing the threshold required to send our checked user mood and checked song from `Module 2` to overwrite our optimal queue building `Module 1`, if required.

B) Training Model

We will be analyzing beats per minute, volume, and lyric Sentiment Analysis on individual users' song's and preferences, and determining what we predict their mood might be. We will use this information to play songs that match the mood of the individual, engaging the individual more and therefore likely increasing the amount of time spent listening to music.

C) Error Correcting Code

One frequent problem that we predict to arise is the changing of moods during the same session for an individual user. A counteractive measure would be to monitor rapid changes in predicted user-mood (**user**.pred_mood) and subsequent rapid lowering in user-mood certainty (**user**.cert_mood). Songs that show rapid transition and lowering of these subsequent values would be shifted to a temporary array, where they will undergo the song module again- asserting that the song's mood has been correctly allotted. Then, the song will be appended to the back of our global queue (our playlist), and previous User-Module values will be reset. The song will undergo our learning algorithm again. Predicted time to remedy instance $O(3^n)$ may seem long,

however, our certainty scores will minimize the amount of songs in the temporary reassignment array (un-ordered deck), thereby having smaller instances of n: resulting in a shorter runtime and lowered potential error. However, we must also point out that as z more possibilities for moods are added, this runtime will increase greatly, as our runtime will be $O((3 + z)^n)$ or $O((Total\ Number\ of\ Mood\ Possibilities)^n)$.

Another potential error would be false matching of moods while the training model is still young. User's will have a wide range of emotions and moods and will all respond differently even when listening to the same songs. This can lead to certain edge cases where our model predicts a user will have a certain mood shift based on the set of data from other tests which in turn would lead to false matching and lower our ML model's AUC. It is important to note that certain mental disorders will also impact mood shifts. Thus, it would be critical to account for this within our model as well to ensure accuracy and specificity. The only way to solve such problems would be to compile a large set of data for our training and validation sets to allow for our model to better handle edge cases and factors such as mental disorders.

VI. Conclusion

A) Model Results

When the user selects his first song to play, the model analyzes the song's lyrics and tempo to create a predicted song-mood pair and an associated certainty. The model adjusts the certainty score based on several attributes including lowering the certainty of the song-mood pair when the song is skipped. For the user, the model predicts his mood based on cached data or current data. Then the model gradually adjusts his mood certainty from the shuffled songs he listens through and the songs he actively selects. With the predicted user mood and associated certainty, the model compares songs in the database to generate an optimized playlist for that individual user, and stores that user's mood matching songs in his individual data for future playlist generation.

With threshold modification, we adjust the threshold value for sending a song from the song module to the user module, or threshold value for adding a song to a user's playlist. We also have error correcting codes to compensate for edge cases and circumstances that could cause our algorithms to make false predictions and incorrect handling of certainties.

B) Interpreting Results

With training, our model is able to fulfill our proposed function. It is able to create an optimized playlist for a user analyzing the positive/neutral/negative mood of the song he listened to and assigning more songs corresponding to his current mood to his playlist. Our model is also

capable of adjusting the playlist when a user changes his mood during his listening session. Our AI model has various applications such as being adopted into music playing software to optimize its shuffle function and improve user experience.

There is a potential drawback of our model. Because we are using the MCI Technique's reverse, the playlist our model created is based on the user's current mood, meaning that the user's mood will determine what songs he will be listening to next. While it might be nothing when the user is in a positive or neutral mood, if the user is currently in a negative mood (depressed, sad, etc), our model's generated playlist may potentially extend the period that his mood stays negative. In other words, a sad person could become sadder listening to the songs, which are played for him based on his sad mood, from his shuffled playlist.

VII. Moving Forward

A) Future Optimization

Possible optimizations include incorporating pulse rate and perspiration data to scan for elevated heart rates or any other signs of excitation or inhibition. This is possible via a smartwatch or other sensors placed on the skin.

Future optimizations could also include subdermal implants, however this may be impractical unless it was already done for other reasons anyways. One possible current use for subdermal audio and mood influence may be to inject subdermal audio emitters for animals in captivity and in zoos, or for the ease of capture-and-release sample collection. Animals are already regularly microchipped in both cases. Often, animals have a different auditory perception frequency range, optimized individually for each animal. Subdermal speakers could be used to play a sound to sooth the animal, and can serve as an alternative to tranquilizer darts or deadly action.

VIII: Referenced Sources

Foster, David; *Hearing and the Vestibular System*; Psychology 110: Introduction to Biological Psychology; *UC Berkeley*, 2021.
Impact Score: Not Available

Coan, James A.; Allen, John J.B.: *Handbook of Emotion Elicitation and Assessment*; *Oxford University Press*, USA; April, 19, 2007;
2020 Impact Score: 3.191

Stratton, Valerie N.; Zalanowski, Annette H.: *Psychology of Music:* Volume 19 Issue 2, page(s): 121-127; *The Pennsylvania State University*: Altoona Campus, 3000 Ivyside Park, Altoona, PA 16601, USA; October 1, 1991.
Impact Score: 2.204

Wagner, David; Weaver, Nicholas; Ada Popa, Raluca; *Computer Security*; *UC Berkeley*, Updated 2021.
Impact Score: Not Available

Spafford, Eugene H.; *Computers & Security: The International Source of Innovation for the Information Security and IT Audit Professional*; ISSN : 0167-4048; Updated October 2021;
Impact Score: 4.438

Owen Yeates, Keith; *Neuropsychology*; ISSN: 0894-4105; January 2020.;
2020 Impact Score: 3.753

Benjamin, Aaron S.; *Journal of Experimental Psychology: Learning, Memory, and Cognition*; ISSN : 0278-7393; First Published : January 2019, Updated September 2021;
2021 Impact Score: 3.435

Coan, Nelson; *Journal of Experimental Psychology: General*; ISSN: 0096-3445; First Published: 2001; Updated : July 2021.;
2021 Impact Score: 5.874