Available online at
https://jcrinn.com/
https://crinn.conferencehunter.com/

Journal of Computing Research and Innovation 9(1) 2024

Journal of Computing Research and Innovation

An AI Chatbot for Personalized Music Recommendations Based on User Emotions

Rula M Ali Farkash[1], Tengku Zatul Hidayah Tengku Petra[2*]

[1]*IT Department, SWIFT Support Services Malaysia Sdn. Bhd., Bangsar South City, Kuala Lumpur, Malaysia*
[2]*School of Computing, College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, Shah Alam, Malaysia*

ARTICLE INFO

Article history:
Received: 31 January 2024
Revised: 22 February 2024
Accepted: 22 February 2024
Online first: 1 March 2024
Published 1 March 2024

Keywords:
AI Chatbot
Deep Learning
IBM Watson
Natural Language Processing

DOI:
10.24191/jcrinn.v9i1.427

ABSTRACT

Most music recommendation systems use data from users' preferences to suggest songs. Popular songs, which have more data, are usually recommended more often, possibly leaving out newer or less popular music. Thus, this study aims to apply machine learning algorithms, such as Deep Learning and Natural Language Processing, to train an AI Chatbot to recommend personalized songs based on user emotions. Firstly, deep learning is employed to predict the mood of individual songs. Subsequently, a new dataset is created based on the predicted mood of each song, which can later be fed into the chatbot to enhance its ability to make song recommendations. Next, the chatbot's intents are defined and integrated into a feed-forward neural network. User messages are analyzed using IBM Watson's natural language analysis function, which returns a sentiment score indicating either a positive, negative, or neutral sentiment. Finally, the chatbot generates a song recommendation from the dataset based on the user's sentiment score and favorite music genre. In this study, two neural network models are developed: one for predicting song moods and the other for training the chatbot. The accuracy results demonstrate that both models achieve high accuracy, scoring 80.4% for predicting song moods and 90% for training the chatbot. These results show that the models are learning effectively and can successfully recommend music based on user emotions.

1. INTRODUCTION

Music plays a vital role in our daily lives. Not only does it serve the purpose of entertainment, but it also offers notable therapeutic benefits. A significant number of individuals have taken advantage of music to achieve emotional healing. A recent study shows that listening to classical instrumental music can help minimize stress and anxiety levels in students and even return cholesterol levels to normal (Bhatnagar et al., 2023). Studies using brain imaging have shown that the right hemisphere is activated when listening to music that relates to the emotional experience, and that even imagining music activates areas on this side of the brain (Blood et al., 1999). According to Davidson (1992, 1998), the right frontal area of the brain is strongly associated with negative emotions, while the left frontal area is highly linked to positive emotions, indicating significant differences between the left and right hemispheres. The overall balance of people's

[2*] Corresponding author. *E-mail address*: tgzatul@uitm.edu.my

https://doi.org/10.24191/jcrinn.v9i1

positive and negative emotions has been shown to predict their judgments of subjective well-being (Diener et al., 1991). Other studies also demonstrated increased activity in brain regions associated with emotion and reward when listening to pleasurable music (Arjmand et al., 2017). The connection between music and emotions is well acknowledged and supported by substantial evidence. However, the significance of this connection is sometimes undervalued within the context of music preferences.

Current music recommendation systems are specifically developed to assist users in effectively navigating and filtering through an extensive collection of music. Nevertheless, it is important to note that these recommendation systems are susceptible to popularity bias, which implies that music tracks with few user interactions are also less likely to receive recommendations. Thus, this study aims to apply machine learning algorithms to train a chatbot on how to interpret user emotions using Natural Language Processing through IBM Watson tone analyzer and suggest songs based on their emotion. The song's mood will be classified using a deep learning algorithm.

Collaborative filtering and content-based filtering are two methods used by current systems to deliver song recommendations. Collaborative filtering assumes that user preferences are a weighted combination of other user preferences, while content-based filtering provides recommendations based on the features that a user desires. Both approaches require large datasets with active users who have previously rated a product before they can generate recommendations. The two major flaws of the above methods can be summarized as follows:

(i) **Cold start problem**: Collaborative filtering relies on the presence of sufficient rating information about users and items. When a new user with no rating history joins the system, it becomes very difficult to initiate high-quality recommendations.

(ii) **Popularity Bias**: Recommender systems are susceptible to popularity bias, where items with more user interactions are more likely to be recommended, while items with lower user interactions have a lower chance of being recommended.

According to the congruency hypothesis, people may pick music to manage their emotions that matches their present emotional state (Grant, 2018). For example, people who have negative moods may choose to listen to sad songs although this seems counterintuitive, but it's shown to improve their happiness in the long term. Other studies (Greenberg & Rentfrow, 2017; Kosinski et al., 2015; Greenberg et al., 2016) has demonstrated that individuals have an attraction for musical works created by artists that possess publicly observable personalities, commonly referred to as "personas," that align with their own unique personality traits. Research made on personality assessments of the artists' and the fans' perceptions revealed patterns demonstrating the relationship between musical tastes and personality (Greenberg et al., 2021).

To investigate this further, Greenberg et al. (2021) employed machine learning techniques and natural language processing to infer the personality traits of individual artists based on their songs. The analysis involved examining the lyrical content of 500 songs performed by 50 artists, with lyrics collected from the ten most popular songs of each artist from azlyrics.com. Utilizing personality prediction models developed by Park and colleagues (2015), the study automatically predicted Big Five personality scores based on the artists' lyrics. The research findings, presented in the paper, demonstrate a substantial correlation between the personality traits of fans and the computer-predicted personalities of the artists, offering empirical support for the real impact of music self-congruity as outlined by Greenberg et al. (2021).

Above studies have shown that people often choose music that matches their emotional state to manage their emotions effectively. Additionally, there is a connection between individuals' musical preferences and personality traits, suggesting that personalized music recommendations based on user emotions could improve user satisfaction and engagement with the system. Therefore, conducting research in this area is essential to develop more effective and personalized music recommendation systems that cater to individual emotional needs and preferences. The main goal of this research is to achieve four specific objectives:

(i) To investigate the effectiveness of various deep learning algorithms in text-based emotion recognition.

(ii) To design a chatbot capable of successfully differentiating between different emotions.

(iii) To develop a personalized song recommendation system tailored to the user's current mood.

(iv) To assess the effectiveness and accuracy of the proposed system.

This paper aims to review machine learning techniques that address the issue of biased music recommendation systems. It is structured into five sections: an introduction, related works, methodology, results and discussion, and conclusion.

2. RELATED WORKS

This section aims to identify the drawbacks in current music recommender systems. To conduct a more efficient and relevant study, a comparison will be made between the proposed system and the most common typical music streaming website, Spotify. Furthermore, the effectiveness of various deep learning algorithms in text-emotion recognition will be analyzed. The findings presented in this chapter contribute to the development of the proposed system, with the goal of delivering improved and more personalized song recommendations to users.

2.1 Recommendation Service in Spotify

According to Dieleman (2014), who worked with Spotify on music recommendation features, Spotify's music recommendation technology is mostly based on collaborative filtering approaches. The goal of collaborative filtering is to estimate users' preferences based on their usage history. For example, consider two users, A and B. If usage data reveals that user A and user B have listened to comparable sets of music, the recommender system can conclude that user A and user B have similar tastes. This approach may also be applied to songs; if two songs are listened to by the same groups of people, they are more likely to belong to the same genre (Porcaro et al., 2022).

There could be limitations when using collaborative filtering for generating recommendations. As noted by Dieleman (2014), a significant hurdle is suggesting less known or new songs. This challenge stems from the fact that collaborative filtering relies on usage data, and popular songs tend to accumulate more usage data. Consequently, the system tends to prioritize recommending popular songs, potentially overlooking newer or less popular music.

While collaborative filtering recommendations are commonly employed in e-commerce and media websites, such as music or movie recommendation platforms, they come with several limitations. First, they require a substantial amount of rating data to be effective; without such data, as in the case of new songs, recommendations cannot be provided. Second, the accuracy of the collaborative recommendation system is closely tied to the number of available ratings (Sánchez-Moreno et al., 2016).

2.2 CRS and Natural Language Processing

The proposed system tackles the limitations of collaborative filtering, such as popularity bias and the cold start problem, by adopting a conversational recommender system (CRS) approach. Initially, a preference framework is developed to determine which questions to ask new users, enabling a quick understanding of their preferences. In a typical CRS, three components are present: a switching mechanism (SWM), a recommendation (REC) module, and a user intention understanding (UIU) module (Sheng et al., 2022). The UIU module manages user inputs and the agent's responses. Given that the most common form of interaction is through text, models have trained a UIU module capable of processing and generating natural language data.

A chatbot is computer software that simulates human communication, enabling people to converse with it as if they were talking to a real person. The proposed system aims to apply machine learning concepts, such as Deep Learning and Natural Language Processing, to train an AI chatbot on how to recommend songs personalized to user emotions. Natural Language Processing (NLP) and machine learning (ML) approaches are employed in text analytics to assign sentiment ratings to themes, categories, or entities within a text. Fig. 1 illustrates the conceptual map of a chatbot using Deep Learning.

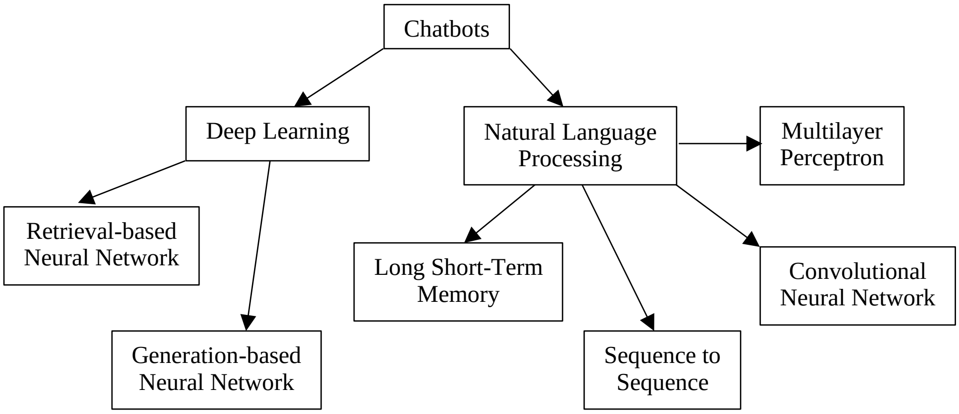

Chatbots
Deep Learning
Natural Language Processing
Multilayer Perceptron
Retrieval-based Neural Network
Long Short-Term Memory
Convolutional Neural Network
Generation-based Neural Network
Sequence to Sequence

Fig. 1. Chat Bot conceptual map using Deep Learning

Source: Bhagwat (2018)

The chatbot should be capable of comprehending the sender's message intentions and deciding the appropriate response type, such as a follow-up question or a direct response. Sentiment analysis can assist the chatbot in gauging the user's mood by analyzing verbal and sentence structure cues.

2.3 RNN’s and LSTM’S in Deep Learning

Recurrent neural network (RNN) and Long Short-Term Memory (LSTM) are special neural network architectures that can process sequential data where chronological ordering matters. A neural network is a method that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain. The primary purpose of a neural network is to analyze data, learn relevant patterns, and apply these patterns to classify new data. Neural networks consist of three sections: an input layer, one or more hidden layers, and an output layer.

Recurrent Neural Networks (RNN) represent a robust type of neural networks, distinguished by their possession of internal memory. With the growing computational power and demand for big data in the current era, along with the introduction of Long Short-Term Memory (LSTM) in the 1990s, RNN has gained prominence. Due to their internal memory, RNN excels at retaining crucial information from the input they receive, making them highly accurate in prediction tasks. LSTM, an enhanced version of RNN, can understand longer data sequences. Like a computer's memory, LSTM can read, write, and delete information from their memory through gated cells. These cells decide whether to store or discard information based on their weight. Weights play a crucial role in helping LSTM determine the importance of information, with three gates in the LSTM system: an input gate for allowing new input, a forget gate for discarding unimportant information, and an output gate that influences the final output based on the importance of retained information.

The importance of research on emotion identification and semantic analysis has expanded as technology has advanced. Emotions are essential components of successful human-computer interaction. Researchers have investigated the usefulness of an LSTM-based deep learning approach for text emotion identification and discovered that, when compared to SVM, Nested LSTM and LSTM give better performance (Haryadi & Kusuma, 2019; Aslam et al., 2022). Nested LSTM gets the best accuracy of 99.17%, while LSTM gets the best performance in term of average precision at 99.22%, average recall at 98.86%, and f1-score at 99.04% (Haryadi & Kusuma, 2019).

Moreover, LSTM performs well in building a Recommender system with specific relevant results. LSTM can forget what it thinks to be not necessary for the long-term. Therefore, the learning results of LSTM are more updatable by time and frequency of interactions. Overall, LSTM is a better model for building the session-based Recommender systems.

2.4 Conversational AI chatbot using Rasa NLU & Rasa Core

In this section, exploration of dialogue handling with Rasa NLU and Rasa Core in the context of an AI Chatbot will be discussed, employing LSTM Supervised Learning and Reinforcement Learning. Natural Language Understanding (NLU), a subset of Natural Language Processing (NLP), utilizes syntactic and semantic analysis of text and speech to discern the meaning of a sentence. The aim of NLU is to extract structured information from user messages, encompassing the user's intent and any entities within their message. While NLU focuses on computer reading comprehension, Natural Language Generation (NLG) facilitates computer-generated writing. NLG involves producing human language text responses based on input data, and this text can also be converted into a speech format through text-to-speech services.

Rasa Core and Rasa NLU, is an open-source Python libraries for creating conversational software, serve as tools for developing a conversational recommender system (Bocklisch et al., 2017). A chatbot equipped with NLG capabilities implies the ability to generate precise and clear responses to user messages. Chatbots typically create responses through dialog management systems or by training prediction models to select appropriate responses based on the context of input messages. Rasa Core represents a Dialog Management solution for NLG.

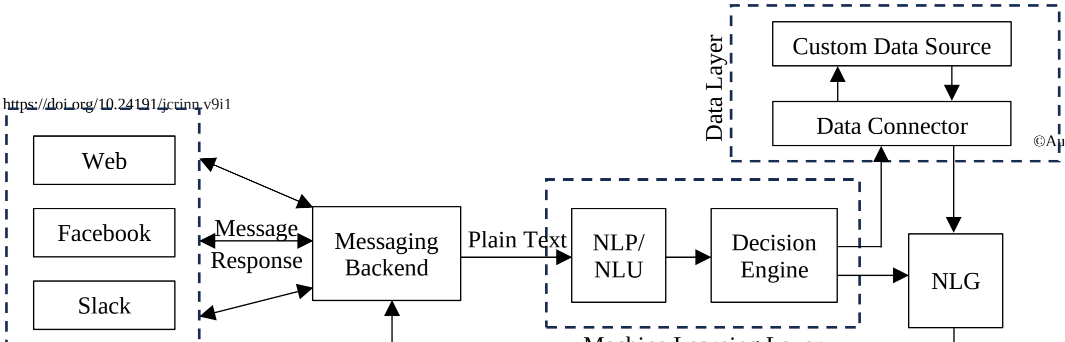


Fig. 2. Conversational chatbots architectural flow in Rasa NLU

Source: Bhashkar (2018)

Rasa NLU comprises loosely connected modules that integrate various natural language processing and machine learning libraries into a unified API. NLU addresses the limited but complex problem of transforming unstructured inputs into structured inputs interpretable and reactive for computers. For instance, if the chatbot prompts the user to state their mood and favorite artist, and the user responds, "I'm feeling sad, and my favorite artist is Coldplay," NLU extracts information such as mood = sad, singer = Coldplay, and action = play song, which the system can comprehend. Additionally, the IBM Watson Tone Analyzer can be employed to read the emotion of each posted message and better identify the user's current mood. Fig. 2 illustrates how NLU and NLG collaborate in a conversational chatbot using Rasa NLU.

LSTM will then map from raw dialog history directly to a distribution over system actions. The LSTM infers a representation of dialogue history automatically, relieving the system developer of most of the human feature engineering of dialogue state. Furthermore, the developer can supply software that describes business rules and gives access to programmatic APIs, allowing the LSTM to perform actions in the actual world on behalf of the user. The LSTM may be optimized via either supervised learning (SL), in which a domain expert gives sample dialogues for the LSTM to copy, or reinforcement learning (RL), in which the system improves by engaging directly with end users.

3. METHODOLOGY

The first section explains the machine learning steps and tools employed in implementing the full working system. It introduces the dataset, explains how the selected features aid in predicting the mood (label) of a song, and explores the utilization of the 'Spotify for web developers' platform and the Spotipy Python library to connect to Spotify’s API and read the audio features of the song tracks. The Keras Python library was used to build the deep learning neural network fitted in an estimator for predicting the label (mood) of a song using the test data. A new dataset will be created based on the predicted mood of each song to later support the chatbot in making song recommendations.

The second section focuses on implementing the chatbot. It explains how the chatbot intents were defined and integrated into a feed-forward neural network. Additionally, the process of loading a new dataset from the previous section into a Python file is detailed for accessibility by the chatbot. The analysis of user messages involves using IBM Watson's natural language analyzer function, which returns a sentiment score indicating positive, negative, or neutral sentiments. The chatbot then suggests a song from the dataset based on the user’s sentiment score and preferred music genre. The final step involves deploying the chatbot into a Flask-based web application for user access to receive song recommendations.

3.1 Song's Mood Prediction

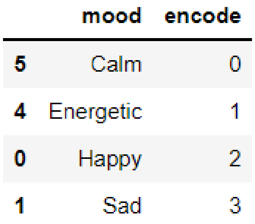
The main goal of machine learning is to design algorithms that automatically assist a system in gathering data and using that data to learn more. Systems are expected to look for patterns in the collected data and use them to make vital decisions for themselves. There are five steps in machine learning to predict the mood of the song:

(i) Understanding Dataset

The dataset for a music track's mood was obtained from Kaggle. The selected features for music mood prediction include Length, Danceability, Acousticness, Energy, Instrumentalness, Liveness, Valence, Loudness, Speechiness, and Tempo. When the dataset is grouped by labels and the mean of the tracks' features is calculated, the findings indicate that the most popular songs are associated with happiness, longer lengths are characteristic of sad songs, energetic songs tend to have a faster tempo, and calm songs are often characterized by their acousticness. Additionally, the most influential indicators of a music track's mood are valence. Tracks with high valence sound more positive (e.g., happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g., sad, depressed, angry).

(ii) Data Preprocessing

This involves cleaning the data to remove unwanted data such as missing or duplicate values and normalizing or scaling the data where appropriate. The initial step involves dropping null values using the dropna() function. The second step is to standardize the dataset, which is crucial to prevent variables measured at different scales from biasing the model. To achieve this, feature-wise normalization, such as Minmax Scaling, will be applied before fitting the model. Subsequently, the 4 mood labels will be encoded into numerical values using dummy encoding as shown in Fig. 3. Once the labels are encoded numerically, the data can be fed into the neural network model. Neural networks require numerical values for both training and testing.

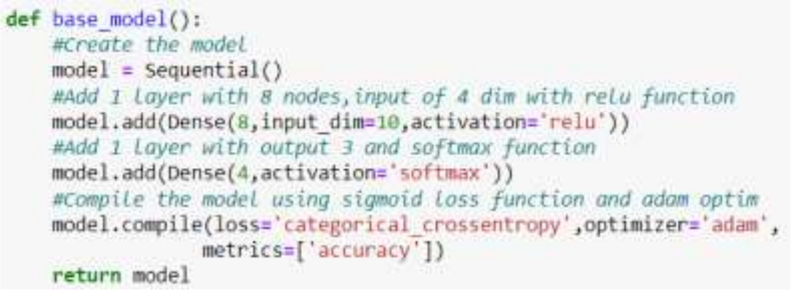


	mood	encode
5	Calm	0
4	Energetic	1
0	Happy	2
1	Sad	3

Fig. 3. Label encoding output

(iii) Build and train the model

The dataset was divided into 80% for training and 20% for testing. The model was built using the Keras library, a high-level deep learning API developed by Google for implementing neural networks. The proposed model is a multi-class neural network with an input layer of 10 features, hidden layer of 8 nodes, and an output layer of 4 labels representing moods (happy, sad, calm, and energetic). To classify the dataset into these moods, the KerasClassifier was employed, taking an argument and a function. The activation function used is Rectified Linear Unit (ReLU), the loss function is a Logistic Function, and the optimizer is the Adam Gradient Descent Algorithm. Fig. 4 shows the details of the Python code to build the model.

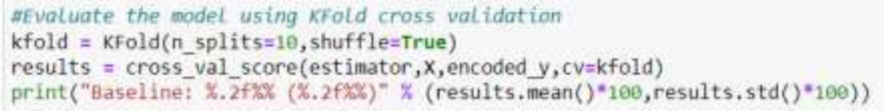


```python
def base_model():
    #Create the model
    model = Sequential()
    #Add 1 layer with 8 nodes,input of 4 dim with relu function
    model.add(Dense(8,input_dim=10,activation='relu'))
    #Add 1 layer with output 3 and softmax function
    model.add(Dense(4,activation='softmax'))
    #Compile the model using sigmoid loss function and adam optim
    model.compile(loss='categorical_crossentropy',optimizer='adam',
                  metrics=['accuracy'])
    return model
```

Fig. 4. Neural network model

(iv) Evaluating the model

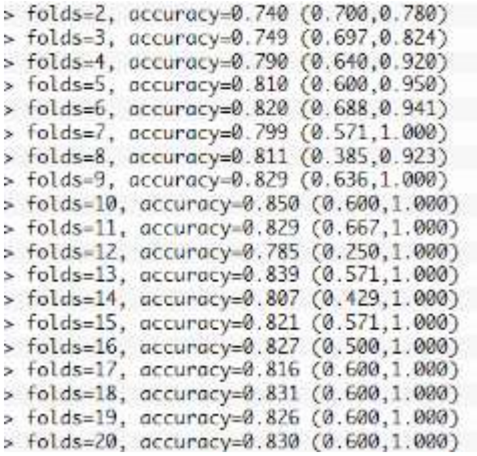
Cross-validation is a resampling procedure employed to assess machine learning models on a limited data sample. Using K-Cross Validation, the dataset is divided into K folds, which are then utilized to evaluate the model's performance when exposed to new data. The python code used to evaluate the model is shown in Fig. 5.



```python
#Evaluate the model using KFold cross validation
kfold = KFold(n_splits=10,shuffle=True)
results = cross_val_score(estimator,X,encoded_y,cv=kfold)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100,results.std()*100))
```

Fig. 5. Evaluating the model

The code will execute a for loop iterating over various k values within folds. The output (Fig. 6) showed that folds=10 achieved the highest accuracy.



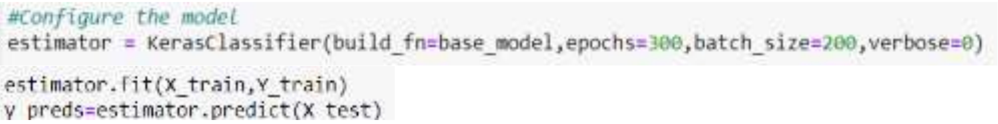
```
> folds=2, accuracy=0.740 (0.700,0.780)
> folds=3, accuracy=0.749 (0.697,0.824)
> folds=4, accuracy=0.790 (0.640,0.920)
> folds=5, accuracy=0.810 (0.600,0.950)
> folds=6, accuracy=0.820 (0.688,0.941)
> folds=7, accuracy=0.799 (0.571,1.000)
> folds=8, accuracy=0.811 (0.385,0.923)
> folds=9, accuracy=0.829 (0.636,1.000)
> folds=10, accuracy=0.850 (0.600,1.000)
> folds=11, accuracy=0.829 (0.667,1.000)
> folds=12, accuracy=0.785 (0.250,1.000)
> folds=13, accuracy=0.839 (0.571,1.000)
> folds=14, accuracy=0.807 (0.429,1.000)
> folds=15, accuracy=0.821 (0.571,1.000)
> folds=16, accuracy=0.827 (0.500,1.000)
> folds=17, accuracy=0.816 (0.600,1.000)
> folds=18, accuracy=0.831 (0.600,1.000)
> folds=19, accuracy=0.826 (0.600,1.000)
> folds=20, accuracy=0.830 (0.600,1.000)
```

Fig. 6. Accuracy score per K Fold

(v) Making Predictions

https://doi.org/10.24191/jcrinn.v9i1

Deep neural networks enable the fitting of complex datasets compared to linear models. However, this also implies a longer training time for the model. The challenging aspect of deep learning lies in fine-tuning all the parameters appropriately. Fortunately, the Tensorflow and Keras estimator's framework provides a structure for organizing data training and evaluation while maintaining the flexibility to experiment with different parameters. Fig. 7 shows the code to make a prediction. *estimator.fit* will train the model with the train data and the *estimator.predict* will predict the model with the test data. The model predicts the mood of songs from Spotify, and the results are listed in the SongsList Excel file.



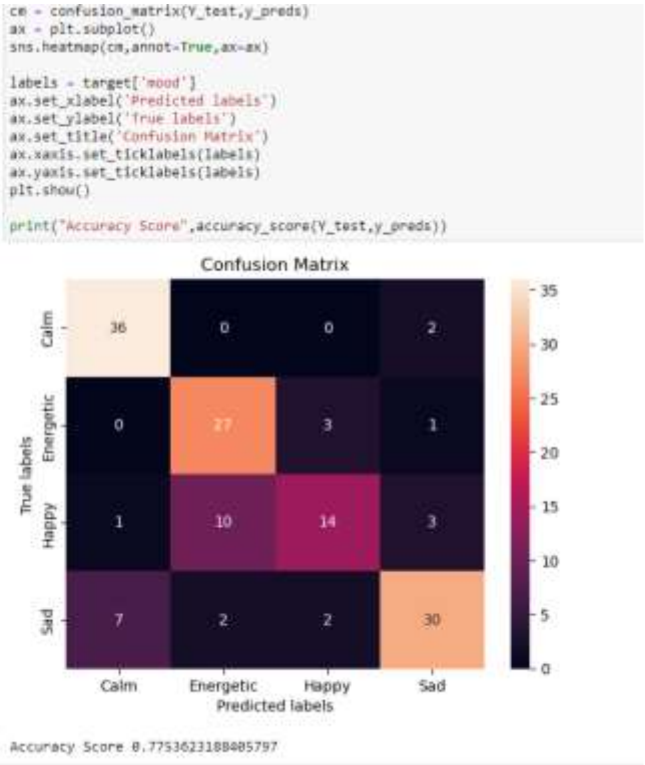
```
#Configure the model
estimator = KerasClassifier(build_fn=base_model,epochs=300,batch_size=200,verbose=0)

estimator.fit(X_train,Y_train)
y_preds=estimator.predict(X_test)
```

Fig. 7. Making predictions

(vi) Evaluating the model performance

A confusion matrix was used to evaluate the performance and accuracy of the model. Utilizing seaborn and matplotlib, the true labels can be visualized against the predicted labels in the matrix as shown in Fig. 8.



```
cm = confusion_matrix(Y_test,y_preds)
ax = plt.subplot()
sns.heatmap(cm,annot=True,ax=ax)

labels = target['mood']
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(labels)
ax.yaxis.set_ticklabels(labels)
plt.show()

print("Accuracy Score",accuracy_score(Y_test,y_preds))
```

Confusion Matrix

	Calm	Energetic	Happy	Sad
Calm	36	0	0	2
Energetic	0	27	3	1
Happy	1	10	14	3
Sad	7	2	2	30

```
Accuracy Score 0.7753623188405797
```

Fig. 8. Confusion matrix and accuracy score (true vs predicted labels)

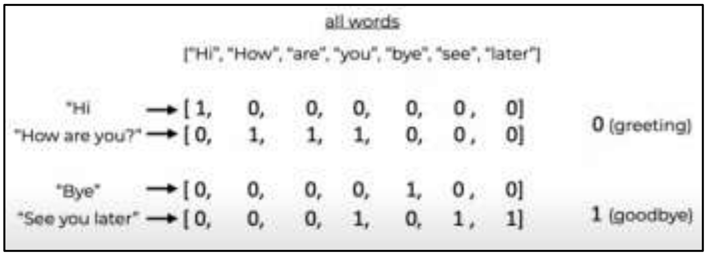
https://doi.org/10.24191/jcrinn.v9i1

3.2 Chatbot Implementation

Rasa NLU is primarily utilized for building chatbots, specifically for tasks like intent classification and entity extraction. To employ Rasa, it requires training data, a set of messages already labeled with their intents and entities. Rasa utilizes machine learning to discern patterns and generalize to unseen sentences. In our project, each intent has various tags (class labels) for different patterns. For instance, the Greetings tag includes phrases like "hi, hello, hey, Good afternoon," each associated with different responses. Below is an example of the Recommend intent in the JSON file:

```
“tag”: “Recommend”,
“patterns”: [
    “Recommend me a new song”,
    “How will you recommend me a song?”,
    “Suggest me a song”,
    “recommend songs”,
    “recommend music”,
    “need new music for my playlist”,
    “need a new song for my playlist”
],
“responses”: [
“Before I can start recommending, what is your favorite music genre? \nPlease type any of the following:\n(pop/rock/rap/classic)”
]
```

Deep learning models cannot be trained with strings, thus, it is necessary to convert the strings into vectors containing numbers. The "bag of words" technique, imported from the Natural Language Toolkit (NLTK), is used for this purpose. Essentially, all the different patterns encountered by the chatbot are split into a single words array known as "all words." The bag of words is employed to create an array for each pattern, which is then compared with the "all words" array. For instance, assuming our "all-words" array is ["Hi", "how", "are", "you", "see", "you", "bye", "later"], and our labels are greetings and goodbyes. If our pattern contains the words "how are you," it will be stored in an array and compared with the "all-words" array. A value of “0” will be assigned if it corresponds to a greeting label, and a value of “1” will be assigned if it corresponds to a goodbye label. Fig. 9 illustrates this concept.



all words

["Hi", "How", "are", "you", "bye", "see", "later"]

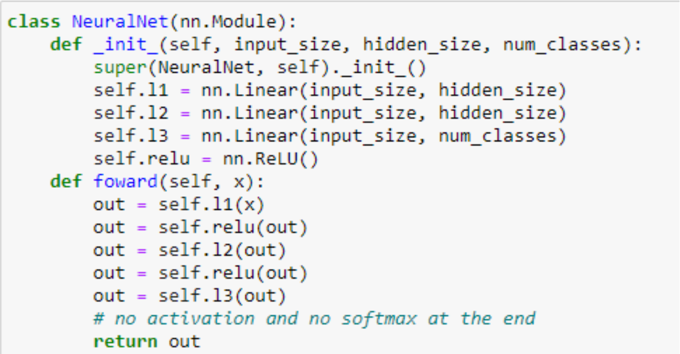
```
"Hi             → [1,  0,  0,  0,  0,  0 ,  0]
"How are you?"  → [0,  1,  1,  1,  0,  0 ,  0]     0 (greeting)

"Bye"           → [0,  0,  0,  0,  1,  0 ,  0]
"See you later" → [0,  0,  0,  1,  0,  1 ,  1]     1 (goodbye)
```

Fig. 9. Bag of Words Concept

The NLTK toolkit is utilized for its comprehensive libraries in tokenization, parsing, stemming, and semantic reasoning. The neural network is created and trained by loading the intents file and establishing three empty arrays stored in a dictionary named "intents." The "all-words" array collects words from the NLU pipeline after tokenization and stemming, while the "tags" array is employed to check if a particular tag matches its corresponding pattern in the intents file.

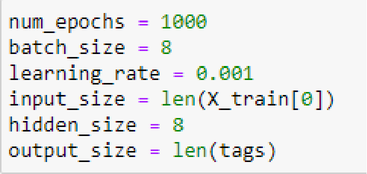
Subsequently, the neural network model is initialized with three linear layers as shown in Fig. 10. The first layer takes the input size and hidden size, the second layer takes the hidden size as both input and output, and the third layer takes the hidden size as input and the number of classes as output. A ReLU activation function is created and assigned to self.relu. The forward path of the neural network is implemented in the def forward function. Each layer is activated using self.relu. Upon creating the model in model.py, it can be imported into the train.py file to commence training. Additionally, bag of words, tokenize, and stem functions are imported from the nltk_utils.py file.



```
class NeuralNet(nn.Module):
    def _init_(self, input_size, hidden_size, num_classes):
        super(NeuralNet, self)._init_()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.l2 = nn.Linear(input_size, hidden_size)
        self.l3 = nn.Linear(input_size, num_classes)
        self.relu = nn.ReLU()
    def foward(self, x):
        out = self.l1(x)
        out = self.relu(out)
        out = self.l2(out)
        out = self.relu(out)
        out = self.l3(out)
        # no activation and no softmax at the end
        return out
```

Fig. 10. Neural network class for chatbot

Next, the hyperparameters are defined as depicted in Fig. 11. The output size represents the different tags, and the input size corresponds to the bag of words.



```
num_epochs = 1000
batch_size = 8
learning_rate = 0.001
input_size = len(X_train[0])
hidden_size = 8
output_size = len(tags)
```

Fig. 11. Hyperparameters

Next, the loss and optimizer are created and set. Loss refers to the loss value of the training data after each epoch. This is what the optimization process attempts to minimize during training, so the lower the value, the better. Accuracy refers to the ratio of correct predictions to the total number of predictions in the training data. Throughout the training process, as shown in Fig. 14 in Result and discussion section, it is observed that the loss decreases, and accuracy increases with each epoch. This indicates an optimal learning curve for our model. The model is trained and stored in a dictionary called "data," which will be accessed later in the chatbot.

During an ongoing chat, the user's message is set as a parameter. The message is tokenized and stored in a variable called "sentence." The "bag_of_words" function is then applied, passing the "sentence" variable and the "all_words" array obtained from the data file. The tag is retrieved by loading the intents.json file and checking if the tag variable matches any of the tags in the intents. If a match is found, a random choice response from the "responses" array associated with the respective tag is returned.

Furthermore, to enhance the chatbot's ability to provide the correct response, a preliminary check is made to ensure that the probability is greater than 0.75 before iterating through the intents. The Softmax function is employed as the output activation function to obtain the probability. Softmax is widely used in

https://doi.org/10.24191/jcrinn.v9i1

deep learning for classification problems and is implemented in the neural network before the output layer. Softmax assigns decimal probabilities to each class in a multi-class problem, and these probabilities must add up to 1.0. This additional constraint aids in training convergence more quickly. The Softmax function formula is as follows:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} for\ i = 1 \ldots\ldots k \tag{1}$$

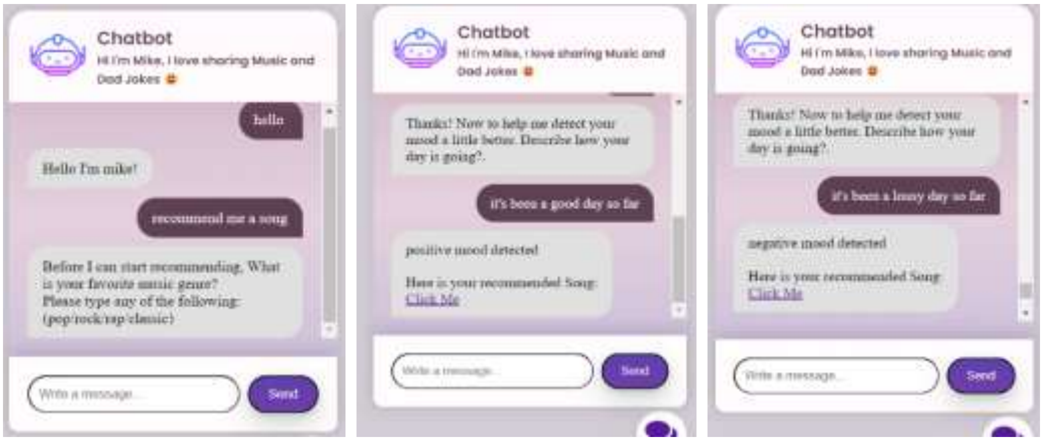
If the probability is less than 0.75, the response will be 'I didn't understand; you can try rephrasing.' In the IBM Watson Natural Language Analyzer function, the text is set equal to the user's message. Emotion and sentiment are set to true in the entities and keywords arrays. The natural language analyzer would throw an error if the text passed to it is too short to be analyzed. To prevent this error, a message is returned to the user if the message sent contains less than 15 characters. This check needs to occur before passing the message to the NLU analyze function. The sentiment score is then inserted into the keywords array. For example, a user's message like "The weather was great today" will return the sentiment label "positive," as shown in the code below:

```
{“usage”: {“text_units”: 1, “text_characters”: 50, “features”: 2},
“language”: “en”, “keywords”: [{“text”: “Today”, “sentiment”;
{“score”: 0.952463, “label”: “positive”}, “relevance”: 0.919812, “count”: 1},
{“text”: “weather”, “sentiment”: {“score”: -.952463, “label”: “positive”},
“relevance”: 0.788983, “count”: 1}], “entities”: []}
```

3.3 Song Recommendation

After implementing the basic flow of how the chatbot communicates with users, additional capabilities need to be incorporated, such as accessing the song lists dataset for providing recommendations. The songList Excel file dataset includes three columns: the song URL, the genre of the song, and the mood of that song. The song URL is passed into the "predict mood" function created in the first section, where the chatbot connects to Spotify's API to retrieve the features of a particular song. After obtaining the mood, the song URL is added to the URL column in the songlist database, along with its predicted mood in the mood column and its genre in the genre column. This process is automated for 500 songs, which will be recommended to users if they match specific columns in the dataset.

When the user requests a song recommendation, the chatbot prompts the user to specify their favorite music genre. Global variables such as Pop, Rock, Classic, and Rap are set to true if the user's message matches the corresponding genre. To analyze the user's current mood, the IBM Watson analyzer function processes the user's message after the chatbot asks the user to describe their day and returns a sentiment score. If a genre variable (e.g., Pop) is set to true and the sentiment label is detected as positive, the chatbot can recommend a random choice from the "pop and happy" array list. This list includes all the column URL values where the mood is happy, and the genre is pop. Similar processes are followed for different genres and mood combinations. Fig. 12 illustrates examples of conversations between a human and the chatbot.



hello

Hello I'm mike!

recommend me a song

Before I can start recommending, What is your favorite music genre?
Please type any of the following:
(pop/rock/rap/classic)

Thanks! Now to help me detect your mood a little better. Describe how your day is going?.

it's been a good day so far

positive mood detected

Here is your recommended Song:
Click Me

Thanks! Now to help me detect your mood a little better. Describe how your day is going?.

it's been a lousy day so far

negative mood detected

Here is your recommended Song:
Click Me

Fig. 12. Examples of conversations between a human and the chatbot

In cases where the user might directly type without explicitly asking the bot for a song recommendation, and the label exists but the genre doesn't, an "else" block is created. In this block, the chatbot returns the user's current mood or label and prompts them to type their favorite music genre if they wish to receive a recommendation.

4. RESULTS AND DISCUSSION

There are two neural network models for evaluation. The first model is the Keras model designed to predict the mood of a specific song, and the second is the feed-forward neural network model developed to train the Chatbot. Hyper-parameter tuning is conducted to configure the models, selecting parameters that yield optimal performance. Finally, the performance of the models is evaluated and measured using the following metrics: Confusion Matrix, Log Loss function, and Accuracy Score.

4.1 Keras Model Training

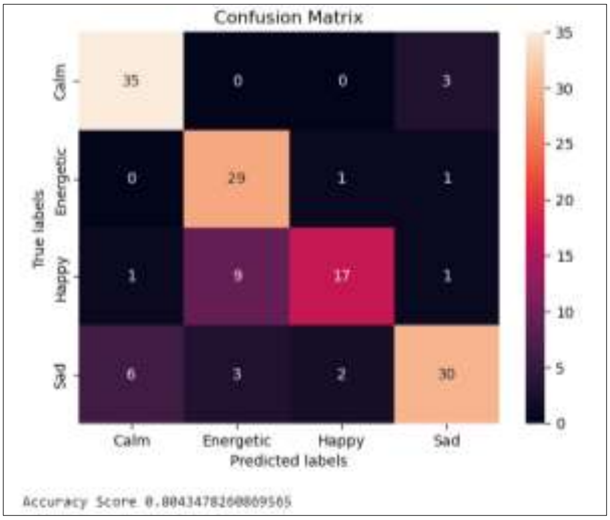
The Adam gradient descent algorithm served as the optimizer for the Keras Neural Network Model. This algorithm optimizes the model during training by identifying features for the most accurate classification. This is achieved by selecting features and weights that minimize the loss of the loss function. Gradient descent is employed to minimize the loss by considering the negative gradient, which corresponds to the rate of the greatest decrease in the loss of the function, until it identifies the parameters leading to a gradient of 0 (minimum loss).

Subsequently, K-fold cross-validation is applied to partition the dataset into K folds to assess the model's ability to handle new data. Setting the number of splits to 10 yielded an average performance metric of 78.71 across the iterations. The model's performance and accuracy are then evaluated using a confusion matrix, resulting in an accuracy of 77.5% (as shown in Fig. 8 before). The outcome indicates that the model performed well in classifying sad songs but encountered challenges in classifying happy and energetic songs. To enhance accuracy, hyparameter tuning is conducted based on the experiment with the dataset. The conducted hyperparameter tuning involves:

https://doi.org/10.24191/jcrinn.v9i1

(i) Batch Size vs Epoch

The number of samples that are processed before the model is changed is known as the batch size. The quantity of complete iterations across the training dataset is the number of epochs. The batch size must be less than or equal to the number of samples in the training dataset. No fixed values or predefined rules exist for selecting the appropriate number of epochs and batch size; it depends on the dataset. Hence, trial and error are necessary to determine the optimal values for our specific problem. In this research, training a Keras Model with one hidden layer, a batch size of 25, and 100 epochs yielded the highest accuracy. After adjusting these parameters, the confusion matrix outputs an improved accuracy of 80.4%, as depicted in Fig. 13.

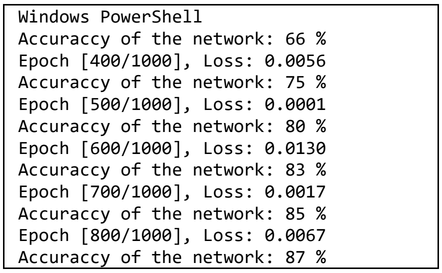


```
Accuracy Score 0.8043478260869565
```

Fig. 13. Confusion matrix after parameter tuning

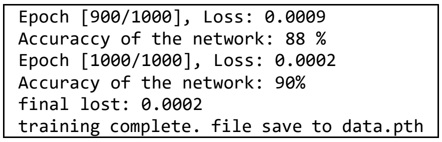
4.2 Chatbot Neural Network Training

The Cross Entropy Loss metric was utilized to assess the performance of the chatbots. Cross entropy loss is a metric employed to gauge the performance of a classification model in machine learning. The loss, or error, is quantified as a number between 0 and 1, with 0 indicating a perfect model. Loss refers to the loss value of the training data after each epoch. Throughout the training, the observed trend shows a decrease in loss and an increase in accuracy as the number of epochs rises (as depicted in Fig. 14). This trend suggests that our model is learning effectively and progressing towards optimal performance.



```
Windows PowerShell
Accuraccy of the network: 66 %
Epoch [400/1000], Loss: 0.0056
Accuraccy of the network: 75 %
Epoch [500/1000], Loss: 0.0001
Accuraccy of the network: 80 %
Epoch [600/1000], Loss: 0.0130
Accuraccy of the network: 83 %
Epoch [700/1000], Loss: 0.0017
Accuraccy of the network: 85 %
Epoch [800/1000], Loss: 0.0067
Accuraccy of the network: 87 %
```

https://doi.org/10.24191/jcrinn.v9i1



```
Epoch [900/1000], Loss: 0.0009
Accuraccy of the network: 88 %
Epoch [1000/1000], Loss: 0.0002
Accuracy of the network: 90%
final lost: 0.0002
training complete. file save to data.pth
```

Fig. 14. Loss and accuracy result

5. CONCLUSION

One of the main challenges encountered during this project was the lack of a powerful GPU, which significantly prolonged the training time of the chatbot's neural network, a resource-intensive task. Moreover, the IBM Watson Natural Language analyzer has limitations; it requires sufficient text input to predict tone accurately, leading to potential inaccuracies with one-word answers. Additionally, inputting integers or text in an unknown language may cause API errors, halting system execution. However, error handlers were implemented to prompt users for clearer input, preventing system crashes. Despite these challenges, deep learning proved effective in independently learning hidden patterns from data. The multi-class neural network model built with the Keras Library accurately predicted song moods using 10 features with 8 nodes and an output layer with 4 mood labels. Optimizing features and parameters, including using the Adam gradient descent algorithm and tuning hyperparameters like number of epochs and batch size, ensured the model's effectiveness in classification tasks.

In future work, we aim to enhance the accuracy of the deep learning model by training it with a larger dataset, conducting more feature engineering, and performing a grid search for the hyperparameters. Grid search is a hyperparameter optimization method that identifies the best combination of hyperparameters for a given model. These combinations, corresponding to a model, are located on a grid, and the objective is to train each model and evaluate it using cross-validation to determine the best-performing model. Additionally, in our Keras multi-neural network song predictor model, the features used to classify the mood label are all sound-related features. This implies that a song's mood is classified based on its sound characteristics, such as its energy, instrumentality, and acoustic properties, rather than the lyrics. For example, a song may have sad lyrics but use energetic and upbeat music, resulting in the model classifying it as happy instead of sad. In the future, we aim to enhance the prediction function to predict the mood of a song based on both lyrical and audio features. The intensity of the emotion conveyed by the song will be determined using the arousal feature, while the valence feature, proportional to the intensity, indicates the positivity of a song.

6. ACKNOWLEDGEMENTS/FUNDING

The authors would like to acknowledge the support of INTI International University, Nilai, Negeri Sembilan for providing the facilities and support on this research.

7. CONFLICT OF INTEREST STATEMENT

The authors agree that this research was conducted in the absence of any self-benefits, commercial or financial conflicts and declare the absence of conflicting interests with the funders.

8. AUTHORS' CONTRIBUTIONS

Rula M Ali Farkash carried out the research and wrote the thesis. Tengku Zatul Hidayah designed the research, supervised research progress, wrote and revised the article.

9. REFERENCES

Arjmand, H. A., Hohagen, J., Paton, B., & Rickard, N. S. (2017). Emotional responses to music: Shifts in frontal brain asymmetry mark periods of musical change. *Frontiers in psychology*, *8*, 2044.

Aslam, N., Rustam, F., Lee, E., Washington, P. B., & Ashraf, I. (2022). Sentiment analysis and emotion detection on cryptocurrency related tweets using ensemble LSTM-GRU model. *IEEE Access*, *10*, 39313-39324.

Bhagwat, V. A. (2018). *Deep learning for chatbots* [Master's Project, San Jose State University]. https://doi.org/10.31979/etd.9hrt-u93z

Bhashkar, K., (2018). Conversational AI chatbot using Rasa NLU & Rasa Core: How Dialogue Handling with Rasa Core can use LSTM by using Supervised and Reinforcement Learning Algorithm. *Medium*. Retrieved from, https://bhashkarkunal.medium.com/conversational-ai-chatbot-using-rasa-nlu-rasa-core-how-dialogue-handling-with-rasa-core-can-use-331e7024f733

Bhatnagar, P., Sachan, A., & Pal, N., (2023). Impact of listening to classical instrumental music on cholesterol levels in Indian medical students. *Int J Acad Med Pharm*, *5*(1), 357-359.

Blood, A. J., Zatorre, R. J., Bermudez, P., & Evans, A. C. (1999). Emotional responses to pleasant and unpleasant music correlate with activity in paralimbic brain regions. *Nature Neuroscience*, *2*(4), 382-387.

Bocklisch, T., Faulkner, J., Pawlowski, N., & Nichol, A. (2017). Rasa: Open Source Language Understanding and Dialogue Management. *ArXiv*. abs/1712.05181.

Davidson, R. J. (1992). Anterior cerebral asymmetry and the nature of emotion. *Brain and Cognition*, *20*(1), 125-151.

Davidson, R. J. (1998). Affective style and affective disorders: Perspectives from affective neuroscience. *Cognition & Emotion*, *12*(3), 307-330.

Dieleman, S. (2014). *Recommending music on Spotify with deep learning*. Published Aug, 5.

Diener, E., Sandvik, E., Pavot, W., Strack, F., Argyle, M., & Schwarz, N. (1991). Subjective well-being: An interdisciplinary perspective. *International Series in Experimental Social Psychology*, *21*, 119-139.

Grant, B. J., (2018). *How current mood state influences song selection behaviors* [Doctoral Dissertation, University of Alabama]. University of Alabama Libraries.

Greenberg, D. M., Kosinski, M., Stillwell, D. J., Monteiro, B. L., Levitin, D. J., & Rentfrow, P. J. (2016). The song is you: Preferences for musical attribute dimensions reflect personality. *Social Psychological & Personality Science*, *7*, 597– 605.

Greenberg, D. M., & Rentfrow, P. J. (2017). Music and big data: A new frontier. *Current Opinion in Behavioral Sciences*, *18*, 50 –56.

Greenberg, D., Matz, S., Schwartz, H. and Fricke, K., (2021). The self-congruity effect of music. *Journal of Personality and Social Psychology*, *121*(1), 137-150.

Haryadi, D., & Kusuma, G. P. (2019). Emotion detection in text using nested long short-term memory. *International Journal of Advanced Computer Science and Applications*, *10*(6), 351-357.

Kosinski, M., Matz, S. C., Gosling, S. D., Popov, V., & Stillwell, D. (2015). Facebook as a research tool for the social sciences: Opportunities, challenges, ethical considerations, and practical guidelines. *American Psychologist*, *70*, 543–556.

Park, G., Schwartz, H. A., Eichstaedt, J. C., Kern, M. L., Kosinski, M., Stillwell, D. J., Seligman, M. E. P. (2015). *Automatic personality assessment through social media language*. *Journal of Personality and Social Psychology*, *108*, 934 –952.

Porcaro, L., Gómez, E., & Castillo, C. (2022). Perceptions of diversity in electronic music: The impact of listener, artist, and track characteristics. In *Proceedings of the ACM on Human-Computer Interaction, 6(CSCW1)* (pp. 1-26). ACM Digital Library. https://doi.org/10.1145/3512956

Sánchez-Moreno, D., González, A. B. G., Vicente, M. D. M., Batista, V. F. L., & García, M. N. M. (2016). A collaborative filtering method for music recommendation using playing coefficients for artists and users. *Expert Systems with Applications*, *66*, 234-244.

Sheng, Q. Z., Zhang, W. E., Hamad, S. A., Khoa, N. L. D., & Tran, N. H. (2022). Deep Conversational Recommender Systems: Challenges and Opportunities. *Computer, 55*(4), 30-39.

https://doi.org/10.24191/jcrinn.v9i1