Assignment 1: Music Genre Classification using a Feedforward Neural Network

This assignment, presented in "Assignment_1.ipynb", outlines the complete workflow for building, training, and evaluating a machine learning model, specifically a **Feedforward Neural Network (FFNN)**, to classify music genres. The core objective is to predict the `Genre` of a song based on its various characteristics.

1. Data Loading and Understanding:

- The assignment begins by loading a dataset named `favorite_music_dataset.csv`. This dataset serves as the raw input for our neural network.
- Each row in the dataset represents a unique song, and it includes several features:
    - `Song_Title`: The title of the song.
    - `Artist`: The artist who performed the song.
    - `Genre`: The musical category (e.g., Pop, Rock, Classical) – this is our target variable that the FFNN will learn to predict.
    - `Release_Year`: The year the song was released.
    - `Duration_Minutes`: The length of the song in minutes.
    - `Listened_Date`: The date the song was listened to.
    - `Platform`: The platform where the song was listened to.

2. Data Preprocessing for Neural Network Input:

- Feedforward neural networks require numerical input and operate most effectively when inputs are on a similar scale. Therefore, significant preprocessing steps are undertaken:
    - **Train-Validation Split**: The dataset is first divided into a training set (75%) and a validation set (25%). The training set is used to teach the FFNN, while the validation set provides an unbiased evaluation of its performance on data it hasn't seen during training, helping to detect overfitting.
    - **Numerical Feature Scaling (Min-Max Normalization)**: Features like `Release_Year` and `Duration_Minutes` are numerical but have different ranges. Min-max scaling transforms these values to a common scale (typically 0 to 1). This is crucial for FFNNs as it prevents features with larger magnitudes from disproportionately influencing the network's weight updates during training, leading to faster convergence and better performance.
    - **Categorical Feature Encoding (One-Hot Encoding)**: Text-based categorical features such as `Song_Title`, `Artist`, `Listened_Date`, and `Platform` cannot be directly fed into an FFNN. One-hot encoding converts these into a binary vector representation. For instance, if there are three platforms (Spotify,

Apple Music, YouTube), each platform would be represented by a unique binary vector (e.g., [1,0,0] for Spotify, [0,1,0] for Apple Music).
    - **Target Variable Encoding (One-Hot Encoding)**: Similarly, the `Genre` (our output/target) is also one-hot encoded. If there are five distinct genres, the FFNN's output layer will likely have five neurons, each corresponding to a genre, and the one-hot encoded target provides the expected binary vector (e.g., [0,0,1,0,0] for the third genre).

3. Feedforward Neural Network Architecture (Implicit):

- While the exact code for the FFNN architecture isn't explicitly detailed in the provided snippet, the use of `import tensorflow as tf` and `model.fit` strongly implies the construction of an FFNN using TensorFlow's Keras API.
- An FFNN typically consists of:
    - **Input Layer**: This layer receives the preprocessed numerical features of a song. The number of neurons in this layer corresponds to the total number of features after one-hot encoding.
    - **Hidden Layer(s)**: One or more intermediate layers where complex patterns and relationships within the data are learned through a series of weighted sums and activation functions.
    - **Output Layer**: This final layer produces the network's prediction. For multi-class classification (like genre prediction), it typically uses a 'softmax' activation function to output probabilities for each possible genre, with the number of neurons matching the number of unique genres.

4. Training the Feedforward Neural Network:

- The FFNN is trained using the `model.fit()` method:
    - **Epochs**: The model is trained for 15 epochs. An epoch signifies one complete pass through the entire training dataset. During each epoch, the network adjusts its internal weights and biases to minimize the difference between its predictions and the actual genres.
    - **Batch Size**: A batch size of 256 is used. This means that instead of updating the network's weights after every single song, the updates occur after processing groups of 256 songs. This balances computational efficiency with the stability of the learning process.
    - **Loss Function and Optimizer (Implicit)**: Although not explicitly stated, during `model.fit()`, a loss function (e.g., categorical cross-entropy for multi-class classification) measures the error, and an optimizer (e.g., Adam, SGD) updates the network's weights based on this error.
    - **Performance Monitoring**: The training output shows the `accuracy` and `loss` for both the training and validation sets at the end of each epoch. The goal is to see validation loss decrease and validation accuracy increase, indicating that the FFNN is learning effectively and generalizing well to new, unseen data.

In essence, this assignment leverages the power of a Feedforward Neural Network to learn intricate relationships between various song attributes and their corresponding musical genres, demonstrating a fundamental application of deep learning in data classification.