# MACHINE LEARNING ASSIGNMENT NO:1

**SUBMITTED TO:** **DR BASHARAT HUSSAIN**

**SUBMITTED BY:** **UMAMA NASEER**

## VIDEO LINK:

https://youtu.be/9zpsMWr9yng

## Dataset Overview

The dataset used for this project is the Spotify music dataset, which contains information about various musical tracks and their features. The dataset includes attributes Here are the attributes in points:

Unnamed: 0

Artist Name

Track Name

Track ID

Popularity

Year

Genre

Danceability

Energy

Key

Loudness

Mode

Speechiness

Acousticness

Instrumentalness

Liveness

Valence

Tempo

Duration_ms

Time_signature

Ratings

The target variable, happy_songs, is a binary classification variable representing whether a song is "happy" or not based on three key features:

Danceability > 0.5

Valence > 0.5

Tempo > 100

This target variable was derived by applying the aforementioned conditions to the dataset.

## Exploratory Data Analysis (EDA)

### 1. Perform data visualization:

Histograms for numeric features.

Scatter plots & correlation matrix.

Boxplots to identify outliers.

### 2. Identify missing values:

There's no missing value in the data set

### 3.Identify Outliers as well

Outliers are identified using the IQR method from boxplot (values outside ( Q1 - 1.5 \times IQR ) and ( Q3 + 1.5 \times IQR )), Z-score method (( |Z| > 3 )), and boxplot whiskers (extreme points)

Popularity - Several high outliers above 80.

Loudness - Many low outliers below -20 dB.

Speechiness - Several high outliers above 0.5.

Tempo - Some low outliers below 50 BPM and high outliers above 200 BPM.

Duration (duration_ms)- Extreme high outliers above 1,000,000 ms.

Liveness - Some high outliers above 0.8.

Time Signature - A few outliers that deviate from the common values (most likely unusual time signatures).

**4.Identify and discuss important features.**

Important features are identified from histogram based on distribution shape, variability, and peaks/skewness.

Popularity - Skewed distribution, indicating most songs have low popularity.

Danceability - Normally distributed, could be useful in predicting song characteristics.

Energy - Right-skewed, with a concentration of high-energy songs.

Loudness - Skewed towards higher values, could correlate with energy and popularity.

Speechiness - Highly skewed with a peak at low values, possibly distinguishing between songs with and without lyrics.

Acousticness - Bimodal, suggesting distinct song types (high vs. low acoustic nature).

Instrumentalness - Skewed, with many songs having low values, suggesting most tracks have vocals.

Liveness - Skewed with a peak at low values, differentiating live performances.

Tempo- Normally distributed, useful for classifying song styles.

Duration (duration_ms) - Highly skewed, with a concentration at lower durations, possibly impacting user engagement.

# Data Preprocessing & Feature Engineering

**1.Convert categorical features to numerical**

In this process i used One-Hot Encoding. It is used to convert categorical features into numerical values (0 and 1) One-Hot Encoding is a technique to convert categorical variables into a format that can be provided to machine learning algorithms. It creates a

binary column for each unique category and assigns a 1 or 0 depending on the presence or absence of that category for a particular instance.

**Identifying Categorical Features:**

Categorical features are variables that represent categories or labels, such as names or types, rather than numerical values.

In the dataset, the following categorical features were identified:

"artist_name"

"track_name"

"track_id"

"genre"

**Applying One-Hot Encoding:**

pd.get_dummies() function was used to apply One-Hot Encoding to the identified categorical columns.

**Result:**

The categorical data was successfully converted into a numerical format that can be used for machine learning models.

**2.Scale numerical features using:**

MinMax Scaling.

I apply MinMax Scaling to normalize numerical features in the dataset, transforming them into the range [0, 1].This ensures that the features have values between 0 and 1.

## Model Selection & Training

Split data into training (80%) and test (20%).

**Train and compare with 4 different models**

- Decision Tree
- Random Forest
- Linear Regression
- Gradient Boosting

**Decision Tree Model :**

Decision Tree Model is used to predict whether a song is "happy" or "sad" based on features such as valence, tempo, and danceability.

Target Variable Creation:

The target variable happy_songs is generated based on certain conditions:

If valence > 0.5

If tempo > 100

If danceability > 0.5

If all these conditions are met, the song is classified as "happy" (1), otherwise "sad" (0).

Feature Selection:

The features used to predict happy_songs are:

valence (happiness level of the song)

tempo (beats per minute)

danceability (how suitable the song is for dancing)

Model Training:

A Decision Tree Classifier is created and trained using the training data (X_train and y_train).

Parameters used for the decision tree:

Criterion: 'gini' (used to measure impurity)

Max Depth: 3 (limits the tree's depth to avoid overfitting)

Prediction and Evaluation:

The model makes predictions on the test set (y_pred), and several performance metrics are calculated:

Accuracy: Proportion of correct predictions.

Precision: How many of the predicted happy songs are actually happy.

Recall: How many of the actual happy songs were correctly predicted.

F1-score: Harmonic mean of precision and recall, balancing both metrics.

The classification report provides detailed metrics for both "sad" and "happy" classes.

Model Visualization:

The trained decision tree is plotted using plot_tree(), providing a visual representation of how the tree splits based on the features.

## 2. Random Forest:

It combines multiple decision trees to make predictions. Each tree is trained on a random subset of the data. SMOTE (Synthetic Minority Over-sampling Technique) is used to oversample the minority class by generating synthetic samples. It helps balance class distributions, especially in imbalanced datasets. Oversampling increases the number of minority class instances, while undersampling reduces the majority class instances (if necessary). SMOTE improves model performance by preventing bias toward the majority class, making the model better at predicting minority class outcomes.

## 3. Logistic Regression

Logistic Regression is a classification algorithm used for predicting binary outcomes (e.g., happy or sad songs). It estimates the probability of the target variable using a logistic function.The model is trained using features and evaluated based on accuracy, precision, recall, and F1-score. It is suitable for situations where the target variable is categorical and relationships between features and target are linear.

## 4. Gradient Boosting:

Gradient Boosting is a machine learning model that combines multiple weak learners (usually decision trees) in a sequential manner to improve predictive accuracy.This code applies Gradient Boosting to predict the happy_songs target variable, evaluates its performance using accuracy, precision, recall, and F1-score, and outputs a classification report.

Evaluation:

Accuracy: The percentage of correct predictions out of all predictions made. It tells you how often the model is correct.

Formula= Corr Prediction/Total Prediction

Precision: The percentage of positive predictions that are actually correct. It tells you how reliable the model is when it predicts a positive outcome.

Formula=TP/TP+FP

Recall: The percentage of actual positives that the model correctly predicted. It tells you how good the model is at identifying positive outcomes.

Formula = TP/TP+FN

F1-Score: The harmonic mean of precision and recall. It balances both precision and recall, especially when the class distribution is imbalanced.

Formula= 2 x Precision+Recall/Precision+Recall

## Hyperparameter Tuning

Hyperparameter tuning was performed using **RandomizedSearchCV** to optimize **Random Forest, Gradient Boosting, and Logistic Regression** models efficiently. A **fast_tune function** was created for quicker searches with **2-fold cross-validation**, using accuracy as the evaluation metric. The best hyperparameters were identified for each model, with **Random Forest performing the best** due to its **high accuracy, ability to handle non-linearity, feature importance, and robustness against overfitting**. This tuning process significantly improved the model's performance and reliability.

## Final Model Evaluation

The confusion matrix is a tool used to assess how well a classification model is performing. It compares the predicted results to the actual outcomes, breaking them down into four categories: true positives, true negatives, false positives, and false negatives.

In this case, the matrix is visualized using a heatmap created by sns.heatmap(), which makes it easier to interpret. The x-axis represents the predicted values, while the y-axis represents the actual values.

I generate the confusion matrix with the confusion_matrix() function, which takes the true labels and the predicted labels as input. This matrix helps identify where the model is making errors.

## Summary & Conclusion

This project focuses on classifying happy songs using machine learning based on key features like danceability, valence, and tempo. The dataset underwent exploratory data analysis (EDA), outlier detection, feature engineering, and scaling to ensure high-quality input for model training. Three models—Random Forest, Gradient Boosting, and Logistic Regression—were trained and evaluated, with Random Forest performing the best due to its robustness, accuracy, and ability to handle non-linearity. Hyperparameter

tuning (RandomizedSearchCV) and SMOTE for class balancing further optimized the model's performance. Final evaluation metrics confirmed that the model effectively classifies songs, making it a reliable tool for predicting song mood based on audio features.