KKBOX's Music Recommendation Challange

In [ ]:

```
# Importing basic libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
import time
import gc
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier, StackingC
lassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.feature_selection import SelectKBest, chi2, f_classif
from sklearn.decomposition import PCA
import xgboost
import lightgbm as lgb
from sklearn import tree
import seaborn as sns
import matplotlib.pyplot as plt
import time
# Deep learning libraries
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, Input, Model
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from sklearn.metrics import roc_auc_score
```

Final Functions

- As we did EDA and FE and apply various ML and DL models on our data points.
- We will define two final functions for pipline.
- First function will take the extracted features and will apply best model on top of it and return the prediction label, either user will listen the song or not.
- Second function will take features and corresponding labels and will return metric.

In [ ]:

```
tr_data = pd.read_csv('/content/drive/My Drive/CS-1/tr_data.csv')
val_data = pd.read_csv('/content/drive/My Drive/CS-1/val_data.csv')
```

In [ ]:

```
def function_1(data_point, best_model):
  '''This function will take features and predict the label using best model'''
  label = best_model.predict(data_point)
  print('The label is : ', label)
```

In [ ]:

```
def function_2(data_point, label, best_model):
  '''This function will calculate metric for given input features'''
  predicted_label = best_model.predict(data_point)
  auc = roc_auc_score(label, predicted_label)
  print('AUC is : ', auc)
```

In [ ]:

```
# sample validation data point for function-1  and function-2
data_point = val_data.iloc[1].drop(['target'])
label = np.array(val_data.iloc[1]['target'])
# load model
best_model = joblib.load('/content/drive/My Drive/CS-1/Data/lgb.pkl')
```

In [1]:

```
function_1(data_point, best_model)
```

```
The label is : [1.0]
```

In [ ]:

```
function_2(data_point, label, best_model)
```

Summary

-> Future steps

- Due to RAM limitations we have taken less amount of data.
- If we use whole data we can get better results.
- Deep learning requires more data to get good results.
- By tweaking parameters on large data points we can achieve better results.
- We can also think of more feature extraction.