1. Design a NLP model on Sarcasm detection.

Electronic journalism powered with Social media has become one of the major sources of information consumption lately. Many media houses are using creative ways in order to tap into increasing views on posts. One of the ways is using sarcastic headlines as click baits. A model that is able to predict whether a piece of headline is sarcastic or not can be useful for media houses in order to analyse their quarterly earnings by strategy. Also, from a reader's perspective, search engines can utilise this information of sarcasm and depending on the reader’s preference, recommend similar articles to them.

The goal is to build a ANN model to detect whether a sentence is sarcastic or not?

https://github.com/Kavitha-Kothandaraman/Sarcasm-Detection-NLP

In [ ]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectional, Drc
from tensorflow.keras.models import Sequential
from sklearn.model_selection import train_test_split

# Load the dataset from the GitHub repository
url = 'https://raw.githubusercontent.com/Kavitha-Kothandaraman/Sarcasm-Detecti
data = pd.read_json(url, lines=True)
# Tokenize the text data
tokenizer = Tokenizer(num_words=10000, oov_token='<OOV>')
tokenizer.fit_on_texts(data['headline'])

# Convert text to sequences
sequences = tokenizer.texts_to_sequences(data['headline'])

# Pad sequences to have the same length
padded_sequences = pad_sequences(sequences, maxlen=100, padding='post', trunca

# Split the data into features (X) and the target variable (y)
X = padded_sequences
y = np.array(data['is_sarcastic'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
model = Sequential([
    Embedding(input_dim=10000, output_dim=128, input_length=100),
    Bidirectional(LSTM(64, return_sequences=True)),
    Bidirectional(LSTM(32)),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_dat
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {test_accuracy}')
```