In [8]:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import pickle

from preprocessor_class import Preprocessor
from functions import f1_metric

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, Input, Dropout, LeakyReLU
from keras.callbacks import EarlyStopping
from keras.utils import to_categorical
from keras.regularizers import l2
from keras.optimizers import Adam
```

Neural Network

This part of the program is largely experimental. I wanted to test the use of a neural network to see if a simple network would perform well on the data. Ultimately, a sequential Dense network didn't do better than my other models and adding additional hidden layers, more epochs, and different transformations didn't help much.

In the interest of time, I decided it was best to focus on the working models instead of looking to improve the neural network, but I am leaving the notebook for the network here so when I have more time, I can experiment further and see if I can get the network to match or exceed the scores of the boosting, bagging, and ensemble models in my final_models.ipynb notebook.

In [4]:

```
data = pd.read_csv('Data/prepared_text_data_sugar.csv')
data
```

Out[4]:

	sugar_class	text
0	5	Cookie Dough Blizzard Cake, 10 in Cookie Dou...
1	5	Reeses Peanut Butter Cups Blizzard Cake, 10 ...
2	5	Chocolate Xtreme Blizzard Cake, 10 in Chocol...
3	5	Oreo Blizzard Cake, 10 in Oreo Blizzard Cake...
4	5	DQ Round Cake, 10 in DQ Round Cake, 10 in DQ...
...	...	...
52926	1	6 Nuggets 6 Nuggets 6 Nuggets, Tenders Entrees
52927	1	Breast, Bonafide Spicy Chicken Breast, Bonafi...
52928	1	Thigh, Bonafide Spicy Chicken Thigh, Bonafide...
52929	1	Leg, Bonafide Spicy Chicken Leg, Bonafide Spi...
52930	1	Black Pepper, for MTO Shnack Wrapz Black Pepp...

52931 rows × 2 columns

In [5]:

```
X = data['text']
y = data['sugar_class']

#One-Hot Encoding target variable, train/test split
ohe = OneHotEncoder(drop='first', sparse=False)
X_train_raw, X_test_raw, y_train_raw, y_test_raw = train_test_split(X, y, test_size = 0.
2, random_state = 200)
y_train_raw = y_train_raw.values.reshape(-1, 1)
y_test_raw = y_test_raw.values.reshape(-1, 1)
y_train = ohe.fit_transform(y_train_raw - 1)
y_test = ohe.transform(y_test_raw - 1)

#Pre-processing and vectorizing text
processor = Preprocessor()
X_train_transformed = processor.fit_transform(X_train_raw)
X_test_transformed = processor.transform(X_test_raw)
vector_pipe = Pipeline([('tfidf', TfidfVectorizer())])
X_train_vector = vector_pipe.fit_transform(X_train_transformed)
X_test_vector = vector_pipe.transform(X_test_transformed)

#Returning independent variables to pd.Dataframe
X_train = pd.DataFrame(X_train_vector.toarray(), columns = vector_pipe['tfidf'].get_feat
ure_names())
X_test = pd.DataFrame(X_test_vector.toarray(), columns = vector_pipe['tfidf'].get_featur
e_names())
```

In [6]:

```
#Ensuring X_train and X_test are proper data types
X_train_array = X_train.values if isinstance(X_train, pd.DataFrame) else X_train
X_test_array = X_test.values if isinstance(X_test, pd.DataFrame) else X_test

y_train_reshaped = np.argmax(y_train, axis = 1)

trainCallback = EarlyStopping(monitor='loss', min_delta = 1e-6, patience = 5)

reg = l2(0.0001)
opt = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07)
```

In [10]:

```
model = Sequential()
model.add(Dense(1800, activation = LeakyReLU(), input_shape = (X_train_array.shape[1],),
kernel_regularizer = reg))
model.add(Dense(4, activation = 'softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer = opt, metrics = f1_metric)
model.fit(X_train_array, y_train, epochs = 15, callbacks=[trainCallback], batch_size= 12
8, validation_split = 0.2)
```

```
Epoch 1/15
265/265 [==============================] - 27s 100ms/step - loss: 5.7495 - f1_metric: 0.4
627 - val_loss: 13.9435 - val_f1_metric: 0.5062
Epoch 2/15
265/265 [==============================] - 27s 103ms/step - loss: 24.2916 - f1_metric: 0.
4824 - val_loss: 34.9437 - val_f1_metric: 0.4615
Epoch 3/15
265/265 [==============================] - 26s 98ms/step - loss: 46.5264 - f1_metric: 0.4
836 - val_loss: 59.1820 - val_f1_metric: 0.3115
Epoch 4/15
265/265 [==============================] - 26s 99ms/step - loss: 71.7045 - f1_metric: 0.4
997 - val_loss: 85.0725 - val_f1_metric: 0.4864
Epoch 5/15
265/265 [==============================] - 26s 98ms/step - loss: 99.0802 - f1_metric: 0.4
704 - val_loss: 113.6235 - val_f1_metric: 0.5096
Epoch 6/15
265/265 [==============================] - 25s 96ms/step - loss: 128.8708 - f1_metric: 0.
4634 - val_loss: 144.1213 - val_f1_metric: 0.4857
```

Out[10]:

```
<tensorflow.python.keras.callbacks.History at 0x243a71f23d0>
```

In [ ]: