horizontal line

**AIL302M - MACHINE LEARNING**

TRAN THANH DUONG

SE160185

Methods to improve accuracy of Viewers' feeling of Youtuber's style Prediction

**24th June 2022**

[COLAB NOTEBOOK](https://colab.research.google.com/drive/1qdNDthzfDo_rHrr475tYCmKolbM5dzI9?usp=sharing)

# INTRODUCTION

In this assignment report, I will try to use some methods with new structure models and modifying features. For more detail, I will you 3 new following methods:

* Features Normalization
* Neural Network and Polynomial Features
* Tree Neural Network Model

# DATA LEAKING FROM PREVIOUS REPORT

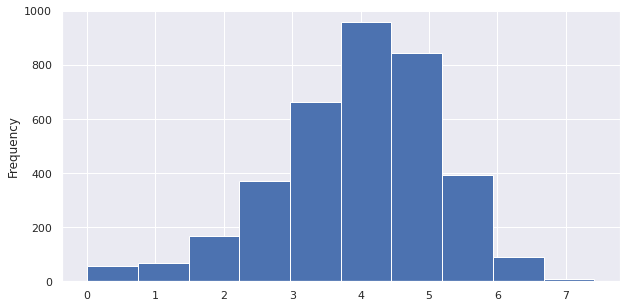
In the previous report, I used **RandomOverSampler** for randomly sampling more data for minority classes (i.e. class 1, class 2, class 3 and class 5). The problem is I sampled data **before** splitting data. This led to data leaking by predicting duplicate data points which misleading us about the accuracy of the model. For example, the class 1 has 170 data points, after resampling data, it increased up to 7000 data points, which is greater than 40 times, which means if one data point is predicted correctly, there are 39 other data points that are predicted correctly, too.

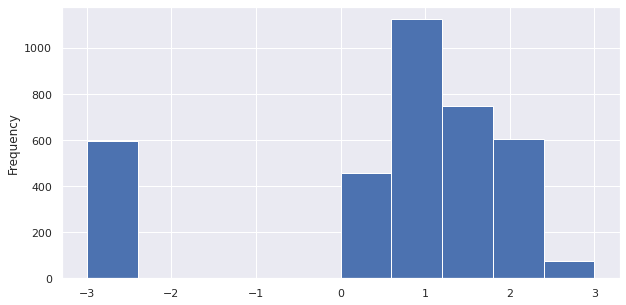
The accuracy/F1-score of testing dataset will just be: **0.36**

# METHODS

## Features Normalization

I will try to convert the features **duration** and **ingre\_count** to fit the normal distribution for better training using logarithms function:





## Model 1: Neural Network with Polynomial Features

**Polynomial Features** is a way to generate more features by multiplying multiple features together. It will help the dataset have more information to train models.

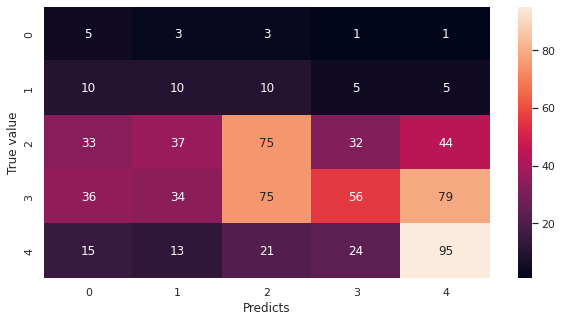
In this method, I will use the interaction between features only with degree 2. The original dataset has 17 features, the number of new features is 153 features (=17+17C2)

After generating the new dataset, I will put the data through the Neural Network with 2 hidden layers (64 nodes, 32 nodes) and 1 output layer (5 nodes).

**Accuracy/Loss/F1-Score on Training and Testing:**

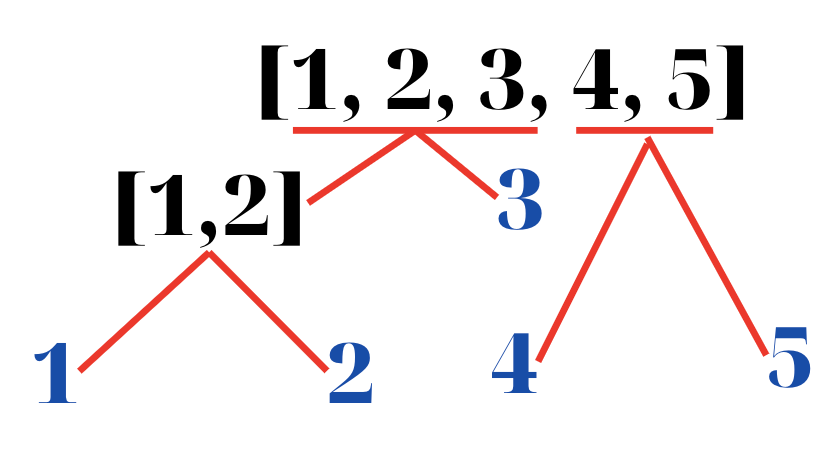
|  | **Accuracy** | **Loss** | **F1-Score** |
| --- | --- | --- | --- |
| **Training** | 0.5946 | 0.9833 | 0.59 |
| **Testing** | 0.3337 | 1.6279 | 0.33 |

**Confusion Matrix**



## Model 2: Tree Neural Network Model

In this method, I will merge the dataset into multiple groups and build a tree model with separate neural networks inside each tree node.



I separate and train 3 different models like above graph:

* Model 1 with 3 labels [1,2,3] → 0, [4] → 1, [5] → 2
* Model 2 with 2 labels [1,2] → 0, [3] → 1
* Model 3 with 2 labels [1] → 0, [2] → 1

The idea behind this method is to prevent unbiased data. The model 1 shows that if we see 3 labels [1, 2, 3] as a single label, it will increase the data points and have a more uniform dataset.

Every model, I will use the same structures of the Neural Network with 1 hidden layer (32 nodes) and the same optimizer and learning rate.

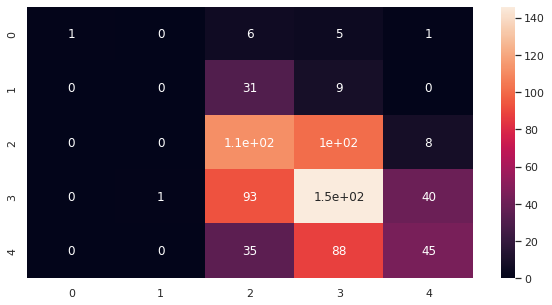
**Accuracy/Loss/F1-Score on Training/Testing in each model:**

|  | **Accuracy** | **Loss** | **F1-Score** |
| --- | --- | --- | --- |
| **Model 1** | 0.4844 / 0.5097 | 0.9915/0.9890 | 0.484 / 0.51 |
| **Model 2** | 0.8051 / 0.8066 | 0.4851/0.5156 | 0.805 / 0.807 |
| **Model 3** | 0.7583 / 0.7736 | 0.5485 / 0.5649 | 0.76 / 0.77 |

**Accuracy/F1-Score on Tree model:**

* Accuracy score: 0.4211
* F1 score: 0.42

**Confusion matrix:**



# CONCLUSION

The accuracy of the model is enhanced in the **Tree Model** (to 0.44 accuracy/F1-score), it shows that the unbalanced data is really mean in training a model and can lead to the bad model if we don't have a solution to analyze the dataset. The accuracy is quite bad in the **Neural Network with Polynomial Features** (0.33 accuracy/F1-score), this can show us that the features we choose do not interact with each other when combined, possibly even leading to a bad model when training on these new generated features.