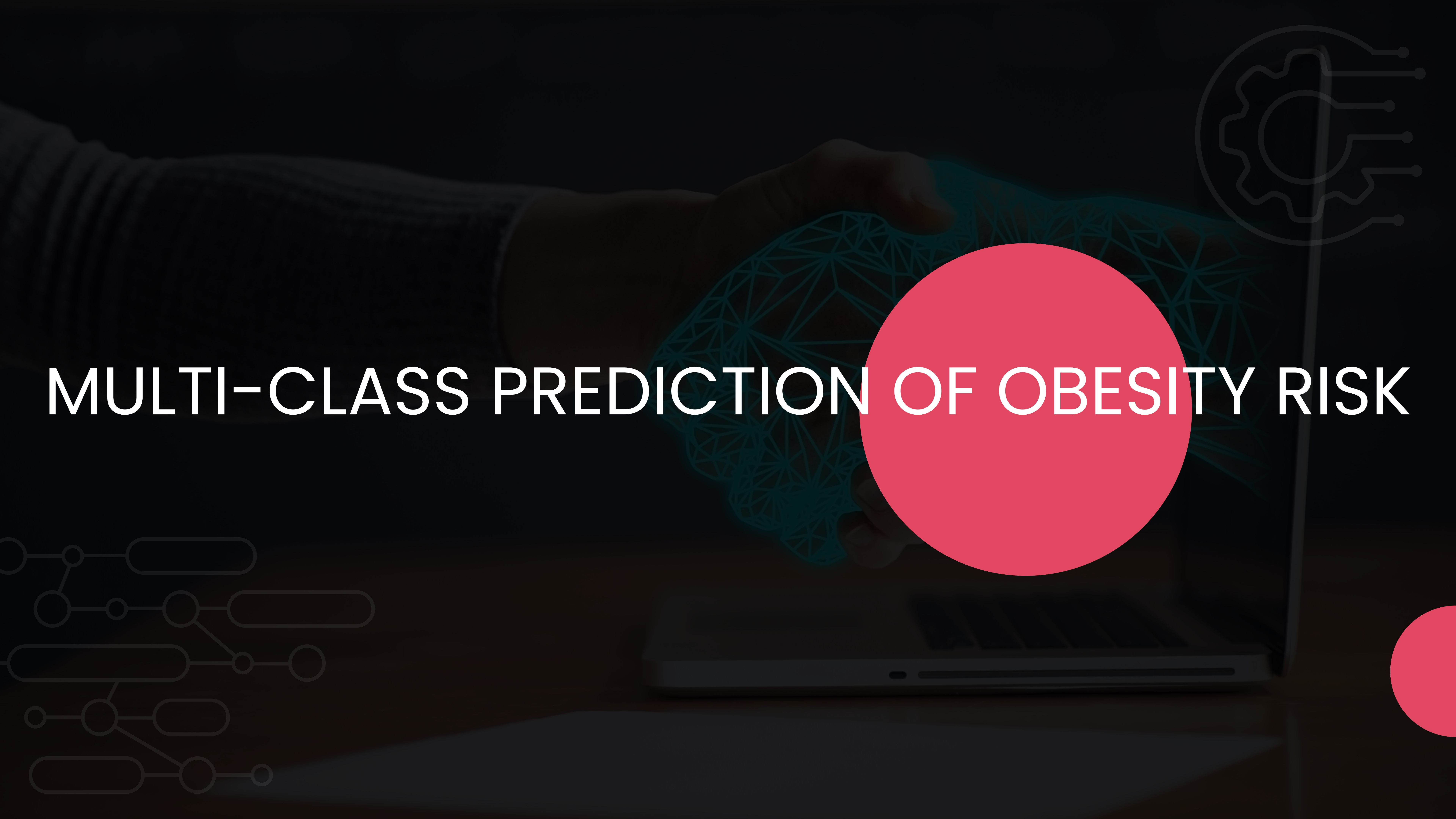
MULTI-CLASS PREDICTION OF OBESITY RISK



PROJECT OVERVIEW

Objective

We aim to develop an ML model which will accurately predict obesity risk among individuals based on various lifestyle factors.

Final Output

Our final project will be able to predict whether a person is at risk of obesity or not based on their lifestyle choices.

PROJECT BRIEF

Our machine-learning model will be able to predict whether a person is at risk of obesity or not. This will be accomplished using a Convolutional Neural Network.

Our system will be trained to predict whether a person will be obese based on factors like BMI, family history, physical activity etc.

https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/

SIGNIFICANCE OF THE PROBLEM

- Obesity is a critical public health issue affecting over 2.1 billion individuals worldwide, constituting approximately 28% of the global population
- It increases the risk of numerous health problems, including heart disease, stroke, type 2 diabetes, and cancers, as well as lesser known complications like sleep apnea and osteoarthritis
- The prevalence of obesity varies notably by region, but has high incidence rates in affluent areas of Europe, North America, and Oceania, and lower rates in less economically developed regions such as South Asia and Sub-Saharan Africa.
- In the United States, obesity rates exceeded 42% among adults by 2020, contributing to the staggering global figure where nearly 40% of adults are either overweight or obese.
- The economic impact is significant. The United States alone incurs about $173 billion annually (in medical expenses related to obesity)

OUR DATASET

01

Source of Dataset

Kaggle:https://www.kaggle.com/competitions/playground-series-s4e2/data

02

Target Feature

NObeyesdad is the categorical target

03

Key Features

Numerical features: Consumption of vegetables, number of meals, height and weight.
Categorical features: Family history, smoking, food consumption between meals and junk food consumption.

04

Why this dataset?

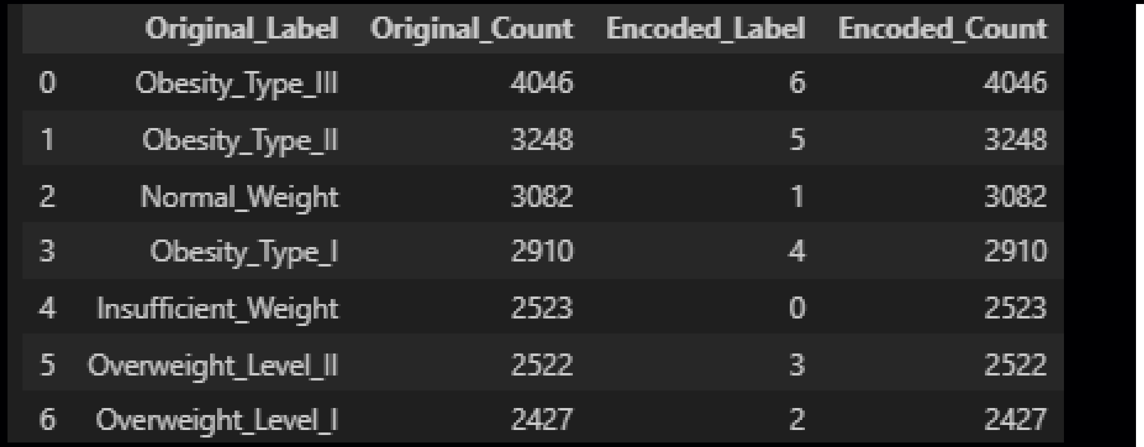
We chose this dataset because it is one of the most comprehensive datasets on obesity. offering diverse features like family history, consumption of vegetables and whole food as well as junk food.

Processing the Target Feature

Type of Target : Categorical
No. of Classes : 7

Encoding Used : Label Encoding

The Target distribution is **Ordinal**, i.e. The classes follow an order and hence can be assigned continuous values, without compromising much with the true meaning.



	Original_Label	Original_Count	Encoded_Label	Encoded_Count
0	Obesity_Type_III	4046	6	4046
1	Obesity_Type_II	3248	5	3248
2	Normal_Weight	3082	1	3082
3	Obesity_Type_I	2910	4	2910
4	Insufficient_Weight	2523	0	2523
5	Overweight_Level_II	2522	3	2522
6	Overweight_Level_I	2427	2	2427

Table: Distribution and Encoding Table for Target

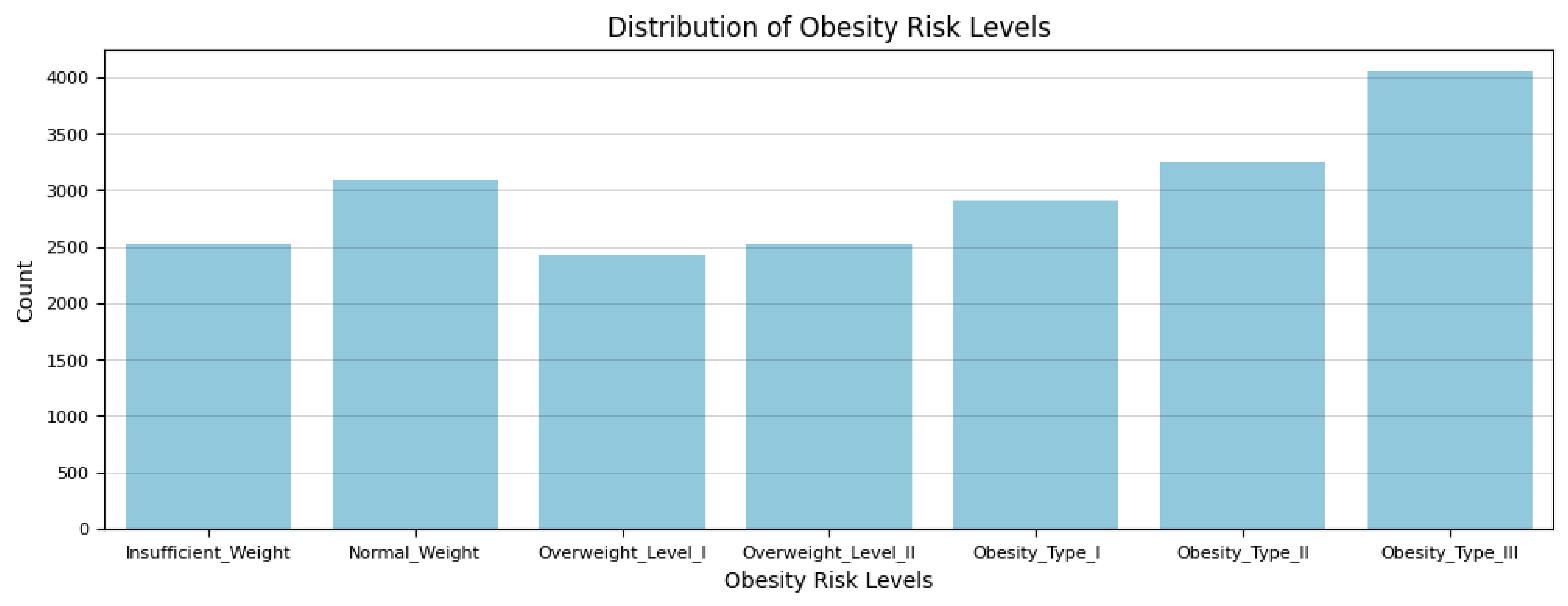


Fig: Distribution of the Datapoints in different Target Classes

Data Pre-Processing

01 Missing/Null Values

Looking for NaN values and interpolating / dropping feature or datapoint.

02 Feature classification

Separating Features in terms of Categorical or Numerical Feature.

03 Feature Encoding

Encoding all Categorical Features into Numerical Features following appropriate Encoding Methods. i.e. Label, One-Hot, Target Encoding.

04 Data type conversion

Assigning appropriate datatypes to the features, while also manipulating numerical features into groups and then encoding.

05 Feature engineering

Analysing features to form informed features that are more relevant to the context of the Target Feature.

05 Feature engineering

Data Pre-Processing

01 Missing/Null Values

Count of Missing / NaN values in :
Training → 0
Testing → 0

∵ No Null Values **No Interpolation** is Required.

02 Feature classification

Number of features in :
Categorical → 8
Numerical → 8

Number of Unique Values in Categorical Features:

Gender	2
family_history_with_overweight	2
FAVC	2
CAEC	4
SMOKE	2
SCC	2
CALC	3
MTRANS	5

Data Pre-Processing

03, 04 Feature Encoding & Data type conversion

Gender	: Ordinal	→ Label Encoding for 1 = 'Male' & 0 = 'Female'
family_history_with_overweight	: Binary	→ Label Encoding for 1 = 'Yes' & 0 = 'No'
FAVC	: Binary	→ Label Encoding for 1 = 'Yes' & 0 = 'No'
CAEC	: Ordinal	→ Label Encoding for 1 = 'Yes' & 0 = 'No'
SMOKE	: Binary	→ Label Encoding for 1 = 'Yes' & 0 = 'No'
SCC	: Binary	→ Label Encoding for 1 = 'Yes' & 0 = 'No'
CALC	: Ordinal	→ Label Encoding for 1 = 'Yes' & 0 = 'No'
MTRANS	: Categorical	→ One-Hot was first considered but, due to heavily skewed nature, we consider the ordinal structure in terms of worse to best option for health with 'Automobile' = 0 & 'Walking' = 4

Grouping Datapoints for certain features :

Age - Generalizing the Population : Ordinal → Label Encoding

Data Pre-Processing

05 Feature Engineering

BMI : Measure of Body Weight Relative to Height

Calorie Intake : The total number of calories you consume daily

Physical Activity : Expenditure of Energy

Water Intake : The total volume of water consumed

Meal Habits : Patterns and behaviors related to how you eat

FEATURE ENCODING

Binary Encoding

- Binary encoding was used on feaures like frequency of consuming vegetables, smoker and SCC.
- We did this because these features have been represented as yes or no and thus are extremely suitable for binary representation.

Ordinal encoding

- Ordinal encoding was on on features like gender, consumption of food between meals, medium of transport and CALC.
- This was done as there are categories in the data which have an ordinal order indicating different intensities like ‘sometimes’, ‘frequently’, etc. This cannot be represnted by just 0 and 1 and thus ordinal encoding was used.

FEATURE ENGINEERING

BMI

- BMI was added as a feature, it was calculated through the general formula for BMI i.e BMI = Weight / (Height 2).
- BMI is a widely used metric to determine if a person is underweight, normal weight, overweight, or obese based on their height and weight.
- We thought that BMI could provide us with very valuable information regarding obesity.

Calorie Intake

- Calorie intake was calculated by summing Frequent consumption of vegetables and number of main meals.
- Total caloric intake for a day helps provide valuable information and is quie relevant in predicting obesity.

Physical Activity

- Physical activity was calculated by subtracting Time Using Technology from Physical Activity Frequency
- Again, physical activity of a person is a massive indicator of their physical health and allows us to infer key whether they have a sedentary lifestyle or not, which holds a strong correlation with obesity.

MODELS USED TO COMPUTE OUTPUT

Decision tree

Decision trees are a machine learning algorithm used for classification and regression. They recursively partition the input space int smaller regions based on feature values, this creates a tree-like structure. Each internal node in the tree represents a decision based on a feature, and each leaf node represents a prediction or class label. Decision trees are much easier to interpret and easily capture non-lnear relationships between features and target variables.

Our decision to use trees was driven by the fact that:

- Their ability to handle non-linear relationships. We have categorical and numerical data which trees can handle extremely well
- Accuracy = 0.850

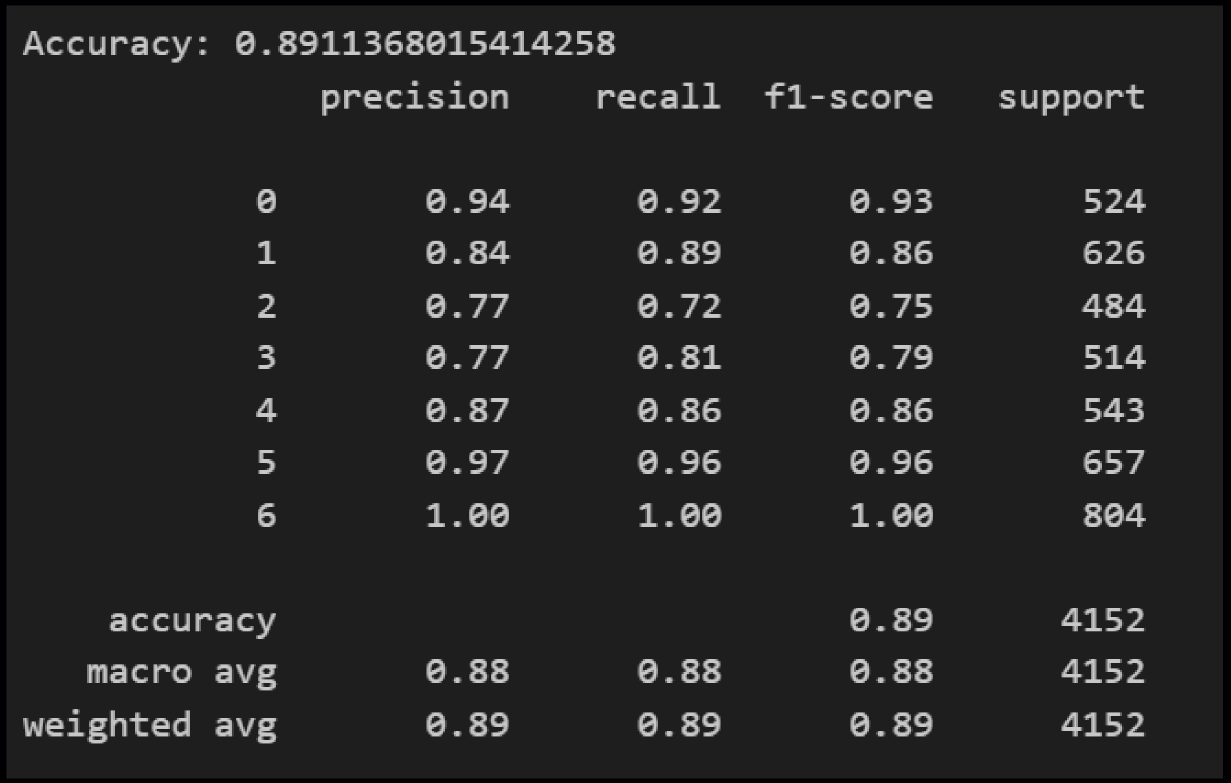
Random forest

Random forests are an ensemble learning technique used for both classification and regression tasks. They recursively partition the input space into smaller regions based on feature values and make a tree-like structure of all the decisions.

They are very easy to interpret and can use both vategorical and numerical data. They are capable of capturing non-linear relationships between features and the target variable.

Our decision to use random forest was driven by the fact that:

- It handles overfitting extremely well.
- Allows us to accurately tune hyperparameters to ensure maximum accuracy
- Accuracy = 0.891



```
Accuracy: 0.8911368015414258
              precision    recall  f1-score   support

           0       0.94      0.92      0.93       524
           1       0.84      0.89      0.86       626
           2       0.77      0.72      0.75       484
           3       0.77      0.81      0.79       514
           4       0.87      0.86      0.86       543
           5       0.97      0.96      0.96       657
           6       1.00      1.00      1.00       804

    accuracy                           0.89      4152
   macro avg       0.88      0.88      0.88      4152
weighted avg       0.89      0.89      0.89      4152
```

MODELS USED TO COMPUTE OUTPUT

Logistic regression

Logistic regression is quite a popular supervised machine learning model which is used for binary classification problems.

It predicts the probability that an input belongs to a certain class x

It uses probabilistic and likelihood maximization techniques to learn.

Decision boundary: Logistic Regression predicts class membership based on a decision boundary.

Applications: Widely used in various fields like in healthcare for disease diagnosis and finance for credit scoring.

We used logistic regression because:

Multi-class classification: Our problem involves multi-class classification which is handled extremely well by logistic regression.

- Accuracy = 0.846

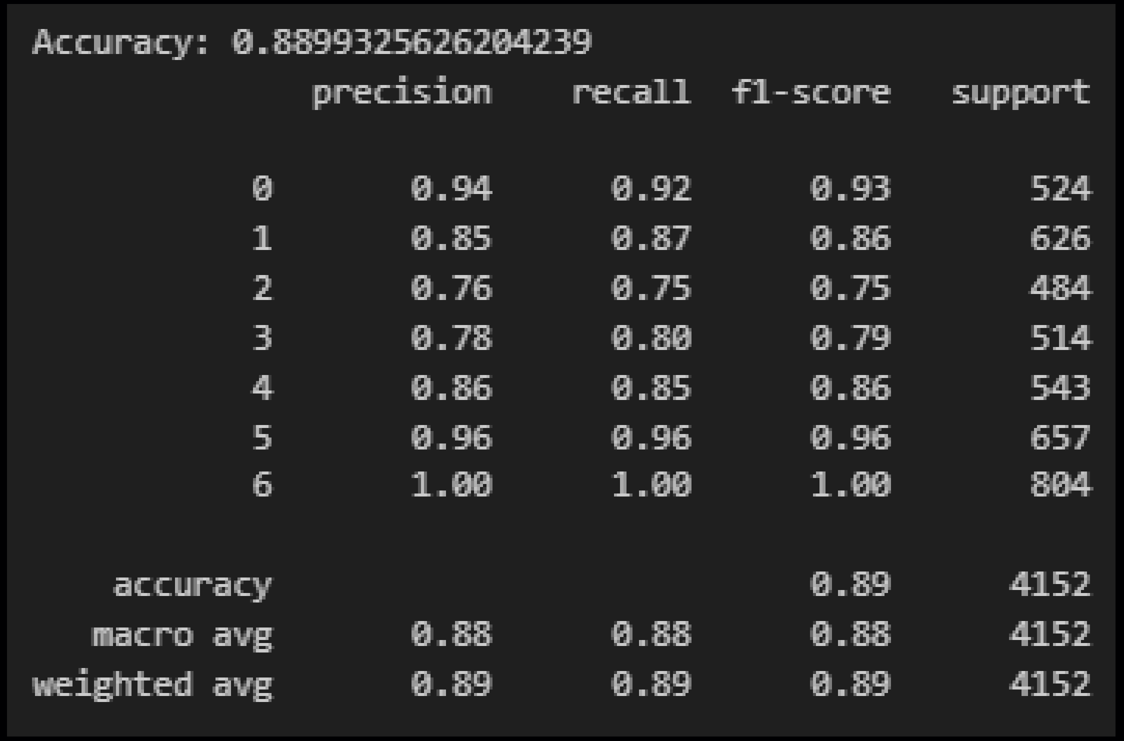
MODELS USED TO COMPUTE OUTPUT

Support Vector Machines

SVM's are very powerful supervised learning algorithms which are used for classification and regression. They work by finding the optimal hyperplane that maximizes the margin between the classes in the training data. SVM's can easily perform non-linear classification by implicitly mapping the input data into a higher dimensional space by using kernels. Thus, SVM's can handle very complex decision boundaries. They are very robust to overfitting and generalize well.

We used SVM's because:

- They are robust to overfitting and can handle our data being multi-dimensional
- Accuracy = 0.864



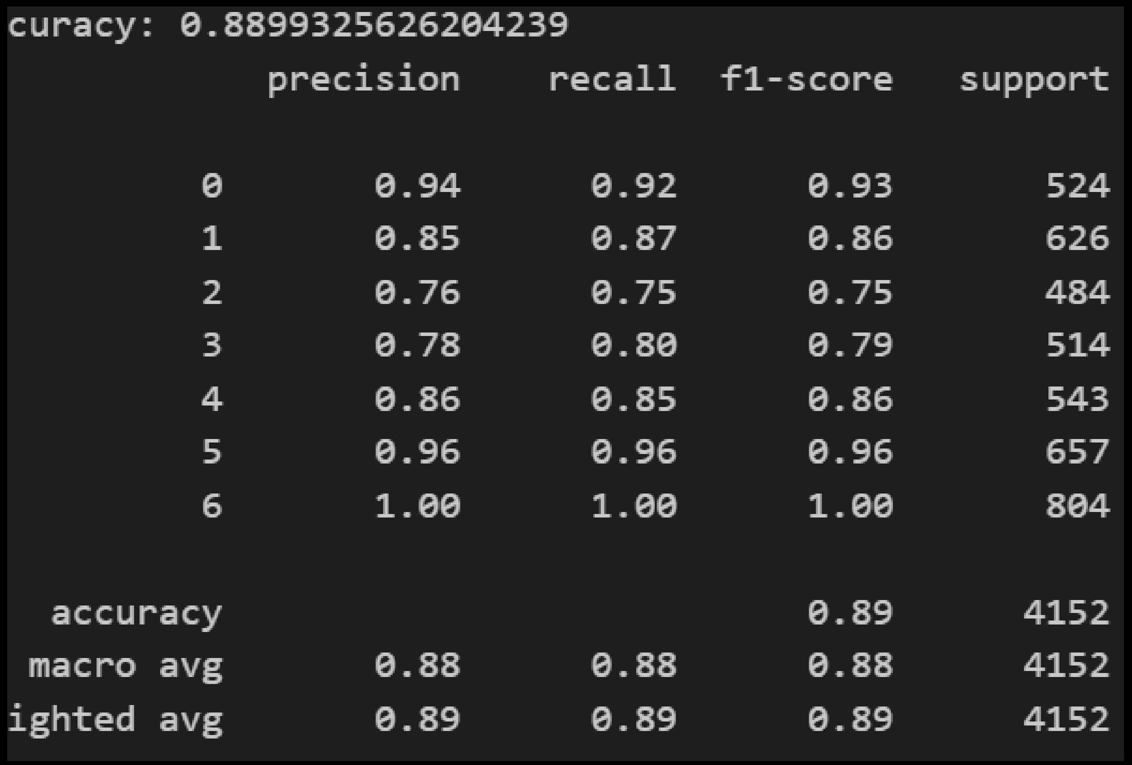
```
Accuracy: 0.8899325626204239
              precision    recall  f1-score   support

           0       0.94      0.92      0.93       524
           1       0.85      0.87      0.86       626
           2       0.76      0.75      0.75       484
           3       0.78      0.80      0.79       514
           4       0.86      0.85      0.86       543
           5       0.96      0.96      0.96       657
           6       1.00      1.00      1.00       804

    accuracy                           0.89      4152
   macro avg       0.88      0.88      0.88      4152
weighted avg       0.89      0.89      0.89      4152
```

XGBoost

XGBoost is a sclaed up and efficient implementation of the grdient boosting framework. In gradient boosting, new models are added whereeach new model corrects errors made by previous models. XGBoost uses several optimizations like parallel and distrbuted computing as well as efficient tree construction and regulaarization to improve performance and reduce overfitting

Our decision to use trees was driven by the fact that:

- Allows us to use GPU acceleration which reduces runtimes.
- Well suited for multi-class classification problems.
- Accuracy = 0.894



```
curacy: 0.8899325626204239
            precision    recall  f1-score   support

         0       0.94      0.92      0.93       524
         1       0.85      0.87      0.86       626
         2       0.76      0.75      0.75       484
         3       0.78      0.80      0.79       514
         4       0.86      0.85      0.86       543
         5       0.96      0.96      0.96       657
         6       1.00      1.00      1.00       804

  accuracy                           0.89      4152
 macro avg       0.88      0.88      0.88      4152
ighted avg       0.89      0.89      0.89      4152
```

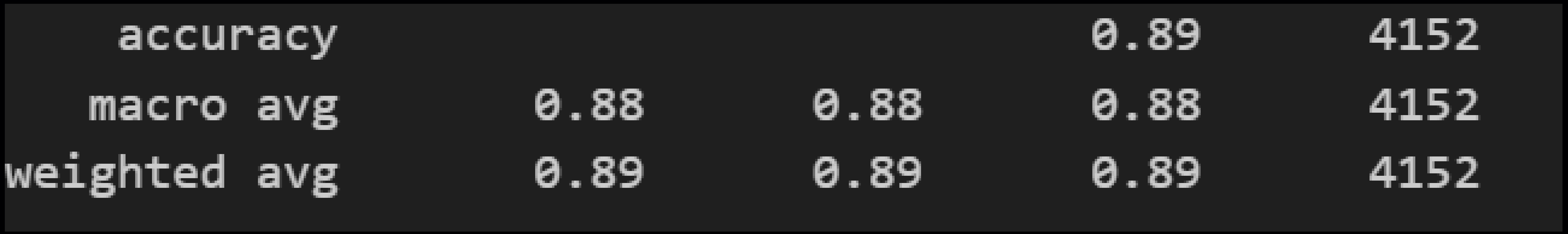
MODELS USED TO COMPUTE OUTPUT

Light Gradient Boosting Decision Tree

LGBM is an implementation of the gradient boosting framework, it is efficient and very good at handling large scale data. It uses histogram based decision tree algorithms, gradient based one-side sampling and oter features to optimize performance and memory use. It supports parallel and GPU based training which makes it useful for large datasets.

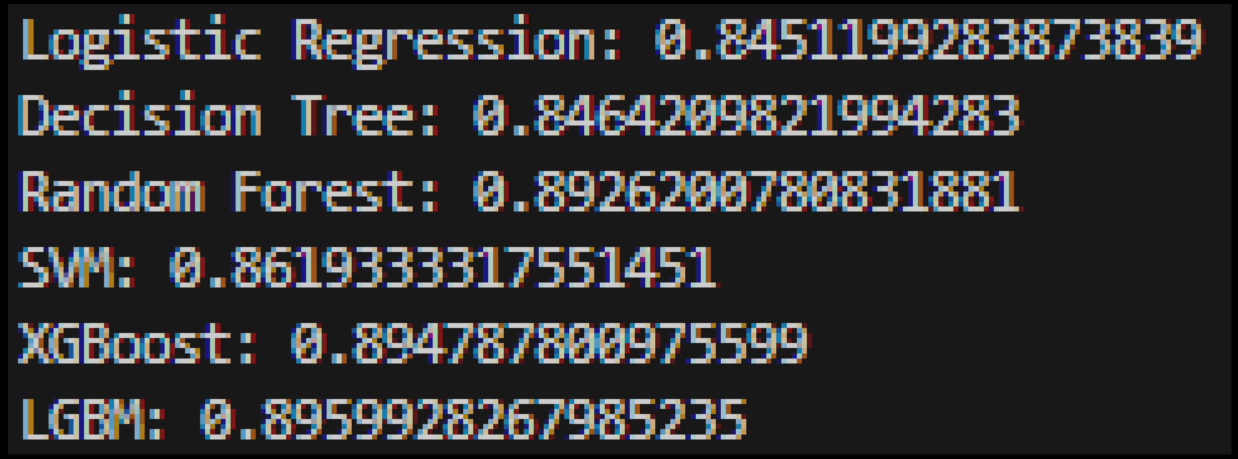
We used LGBM due to it’s ability to:

- Efficiently handle multi-class problems
- It provides support for GPU based learning which we are leveraging
- Better accuracy than other boosting algorithms and handles overfitting better.
- Accuracy = 0.892



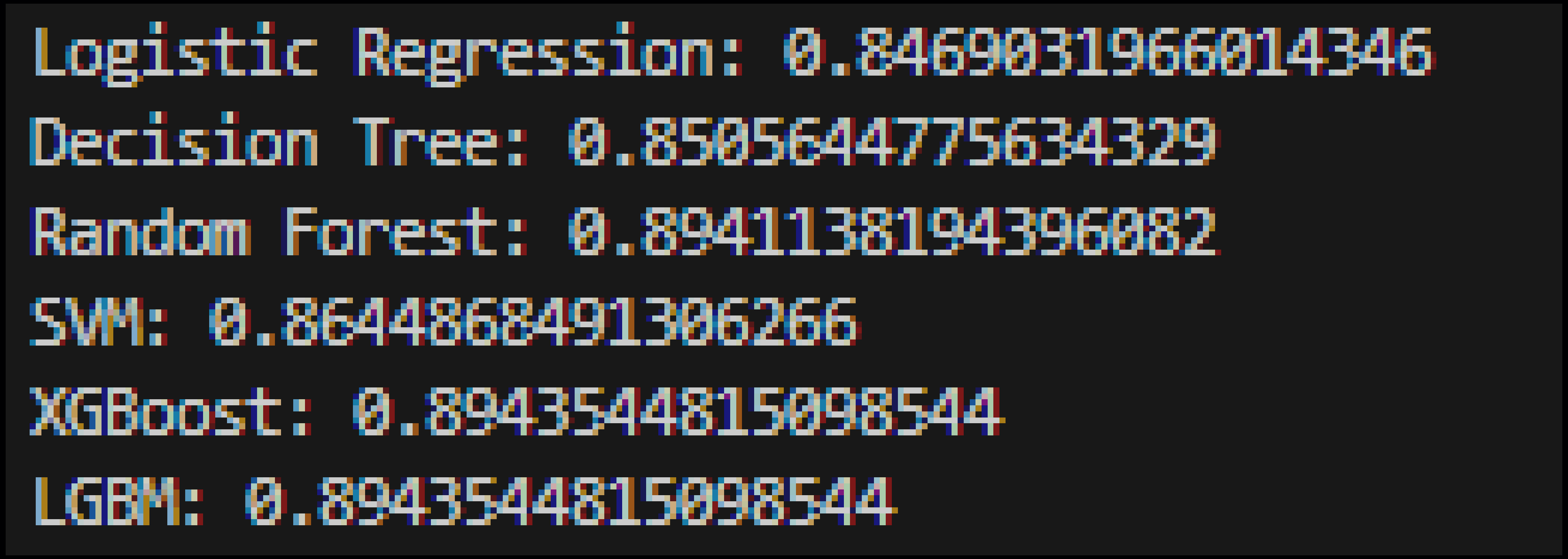
```
    accuracy                           0.89      4152
   macro avg       0.88      0.88      0.88      4152
weighted avg       0.89      0.89      0.89      4152
```

K-Fold CV



```
Logistic Regression: 0.8451199283873839
Decision Tree: 0.8464209821994283
Random Forest: 0.8926200780831881
SVM: 0.8619333317551451
XGBoost: 0.894787800975599
LGBM: 0.8959928267985235
```

K-Fold CV for k=5



```
Logistic Regression: 0.8469031966014346
Decision Tree: 0.8505644775634329
Random Forest: 0.8941138194396082
SVM: 0.8644868491306266
XGBoost: 0.8943544815098544
LGBM: 0.8943544815098544
```

K-Fold CV for k=10

CONVOLUTIONAL NEURAL NETWORK

We used a CNN due to the fact that the CNN architecture is designed to extract all the relevant features from a person’s life and use it to calculate convolutional and pooling layers and hence provide accurate predictions.

- 1-D Convolution layers
  - These provide 1-D convolutions which are useful for analyzing sequential data, which in our case is the test data which includes all our feature engineering.
  - **BatchNormalization:** Followed by a batch normalization layer which normalizes the activations from the previous layer at each batch, maintaining the mean activation close to 0 and the activation standard deviation close to 1.
  - MaxPooling1D: This is followed by a max pooling operation with a pool size of 2, reducing the spatial dimensions (i.e., the length of the sequence data).
  - Dropout (0.3): A dropout layer with a rate of 0.3 is used to prevent overfitting by randomly setting a fraction of input units to 0 during training.
  - 2nd 1-D Convolution Layer has 128 filters and a Dropout Rate of 0.4

- LeakyReLU Activation
  - Variant of the ReLU function which helps mitigate the vanishing gradient problem.
  - Alpha value of 0.01 allows for better gradient flow.

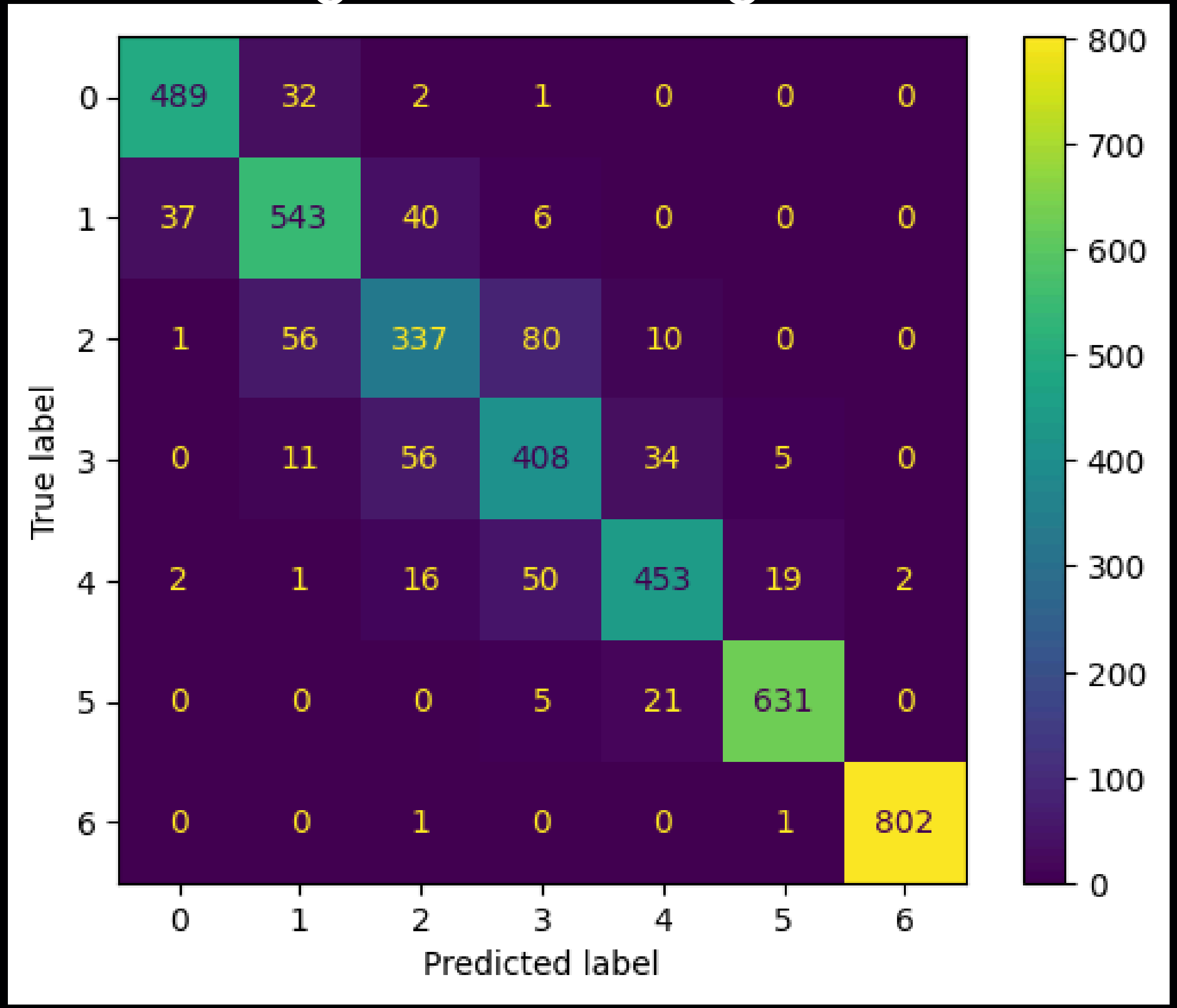
- MaxPooling1D Layers
  - Our pool size of 2 downsamples the input into 1D region and computes the maximum of them all.

- Dropout Layers
  - Dropout Layer rate of 0.3 means that 30% of the inputs are randomly set to 0 which helps reduce overfitting

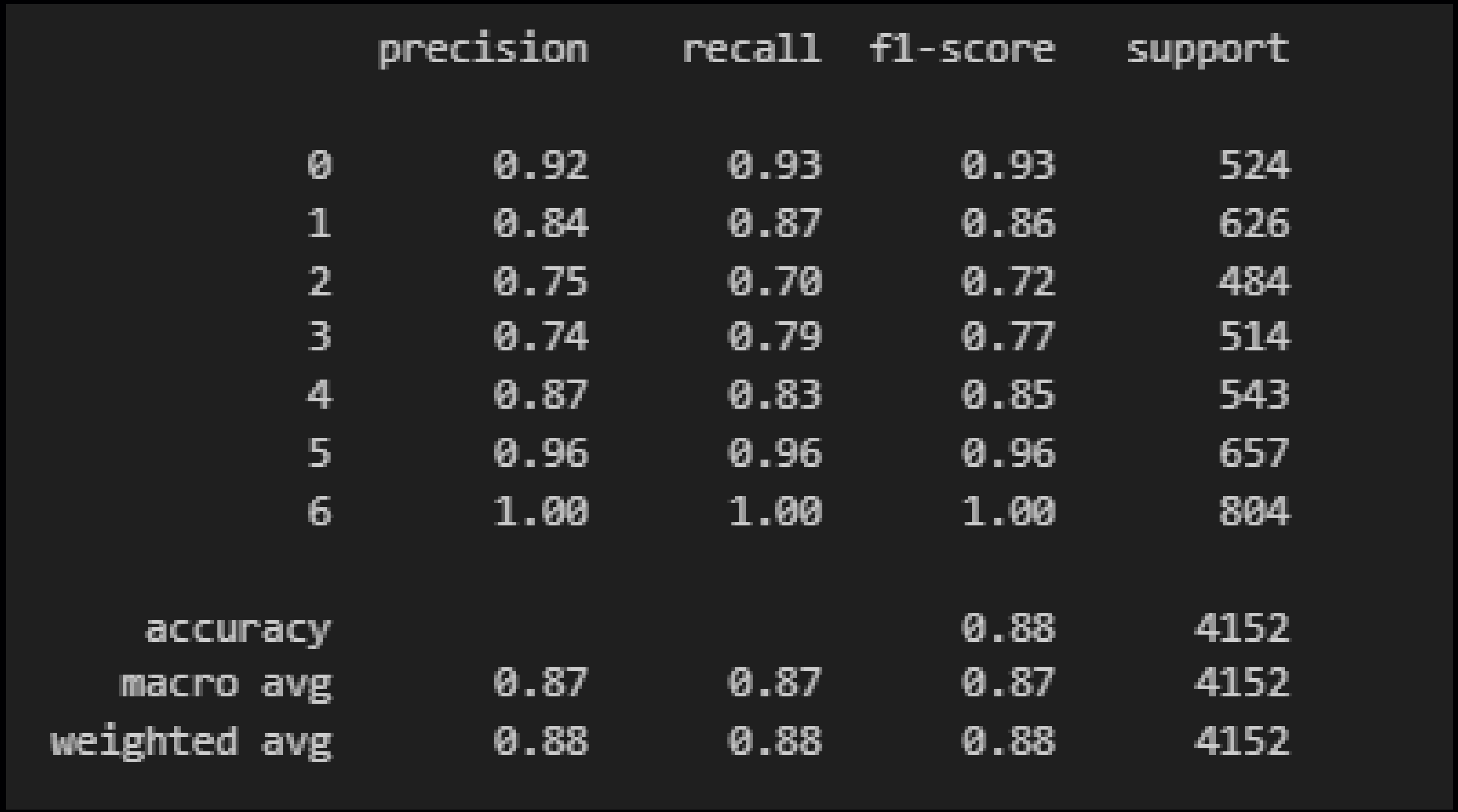
CONVOLUTIONAL NEURAL NETWORK

- Flatten Layer
  - Converts the multi-dimensional output to a 1-D vector

- Dense Layers
  - A fully connected layer with 128 neurons, also using the ReLU activation function. This is followed by a dropout layer with a rate of 0.5 to further prevent overfitting.
- Output Layer: The final layer is a dense layer with a number of neurons equal to the number of unique classes in the target data, using the softmax activation function to output probabilities for each class.



	0	1	2	3	4	5	6
0	489	32	2	1	0	0	0
1	37	543	40	6	0	0	0
2	1	56	337	80	10	0	0
3	0	11	56	408	34	5	0
4	2	1	16	50	453	19	2
5	0	0	0	5	21	631	0
6	0	0	1	0	0	1	802



```
              precision    recall  f1-score   support

           0       0.92      0.93      0.93       524
           1       0.84      0.87      0.86       626
           2       0.75      0.70      0.72       484
           3       0.74      0.79      0.77       514
           4       0.87      0.83      0.85       543
           5       0.96      0.96      0.96       657
           6       1.00      1.00      1.00       804

    accuracy                           0.88      4152
   macro avg       0.87      0.87      0.87      4152
weighted avg       0.88      0.88      0.88      4152
```

THANK YOU