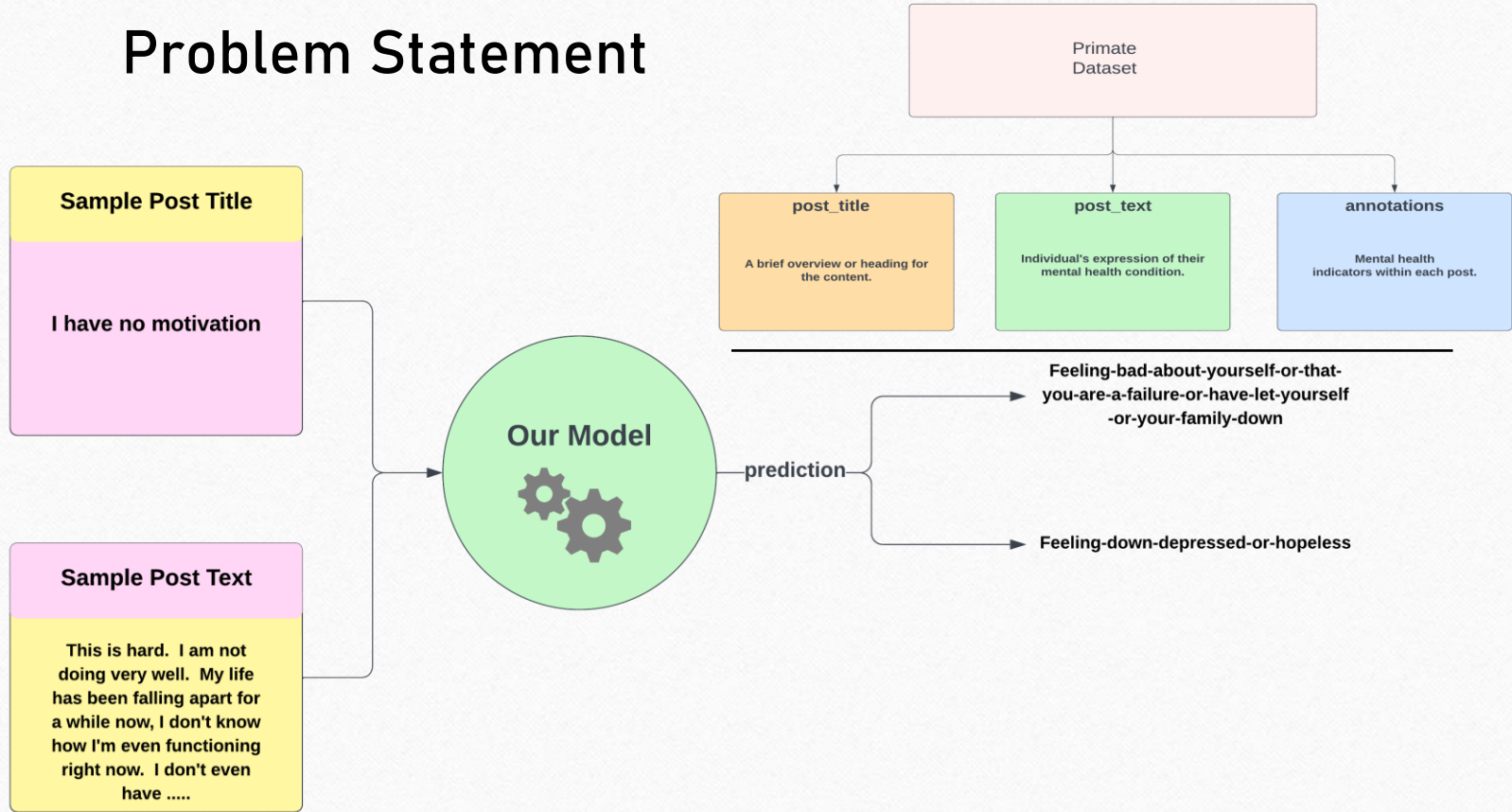# TEAM DETAILS AND PROBLEM STATEMENT

- **Problem Statement** : categorizes paragraphs based on the dataset's presence or absence of specific mental health indicators.
- **Team Name :** bHUC

- **Team Leader Details (Name, Phone Number, Email) :** Drishtant Singh Sengar , 9044400879 bt22cse012@iiitn.ac.in

- **Institution Name :** IIIT Nagpur

- **Course Enrolled :** Btech CSE

Problem Statement

Observations : Class imbalance

Distribution of annotations
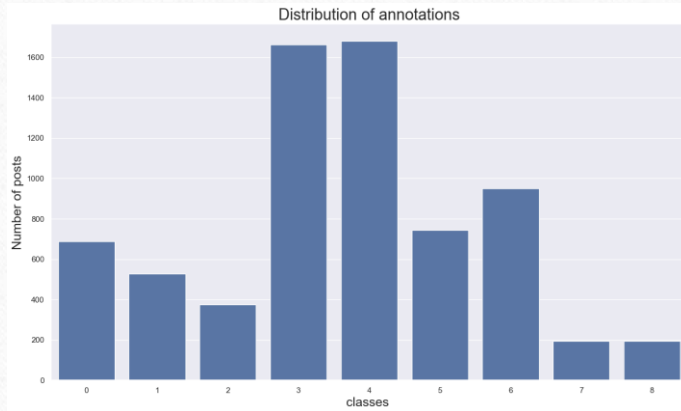
1.Imbalanced Classes: When the classes are imbalanced, meaning some classes have significantly more instances than others, the model may become **biased towards the majority class**. This can lead to **poor performance** on the minority classes and inaccurate predictions.

2.Limited Representation: Annotations with fewer instances may not have enough data to accurately learn their patterns and characteristics. As a result, the model may struggle to make accurate predictions for these annotations.

3.Evaluation Metrics: When evaluating the model's performance, metrics such as **accuracy may not provide an accurate representation of its effectiveness**. For imbalanced datasets, metrics like precision, recall, and F1-score are more informative in assessing the model's performance on individual classes.
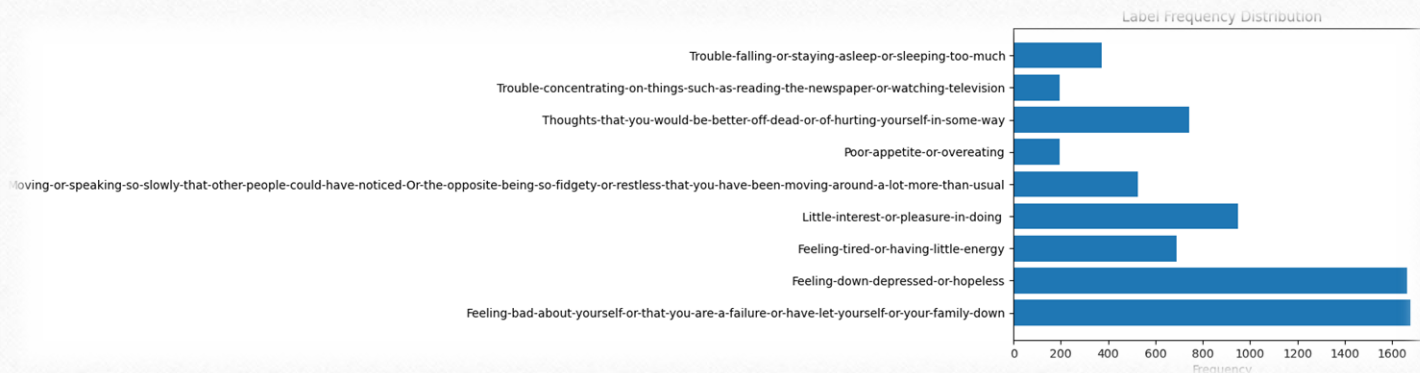
Our Approach

To solve the problem of **label/class imbalance** we assigned different **weights** to each label to improve the precision and recall of our model.

```
sample_weights = sample_weights[:-401]
sample_weights
```
✓ 0.0s

```
array([0.27509327, 0.07394186, 0.09504036, ..., 0.07394186, 0.26618515,
       0.32206228])
```

Label Frequency Distribution

Our workflow(addressing class imbalance)

```
['Feeling-down-depressed-or-hopeless', 'no'],
['Feeling-tired-or-having-little-energy', 'yes'],
['Little-interest-or-pleasure-in-doing ', 'yes'],
['Moving-or-speaking-so-slowly-that-other-people-could-have-noticed-Or-the-opposite-being
 'no'],
['Poor-appetite-or-overeating', 'no'],
['Thoughts-that-you-would-be-better-off-dead-or-of-hurting-yourself-in-some-way',
 'no'],
['Trouble-concentrating-on-things-such-as-reading-the-newspaper-or-watching-television',
 'no'],
['Trouble-falling-or-staying-asleep-or-sleeping-too-much', 'no']]
```

Binarizer

```
array([[1, 0, 1, ..., 0, 0, 0],
       [1, 1, 0, ..., 0, 0, 0],
       [1, 1, 1, ..., 1, 0, 0],
       ...,
       [1, 1, 0, ..., 0, 1, 0],
       [1, 1, 1, ..., 1, 0, 0],
       [0, 1, 0, ..., 0, 0, 0]])
```

**encoded_labels**

sample weights

```
array([0.27509327, 0.07394186, 0.09504036, ..., 0.24478142, 0.08557239,
       0.13731608])
```

computing
sample weights

These are extremely black boxed explanations of the ML models to
know how they were really implemented check out the Jupyter notebooks here!

Our workflow(vectorizing data)

Post Text Content

Processing

stemming & stopwords

TF-IDF Vectorizer
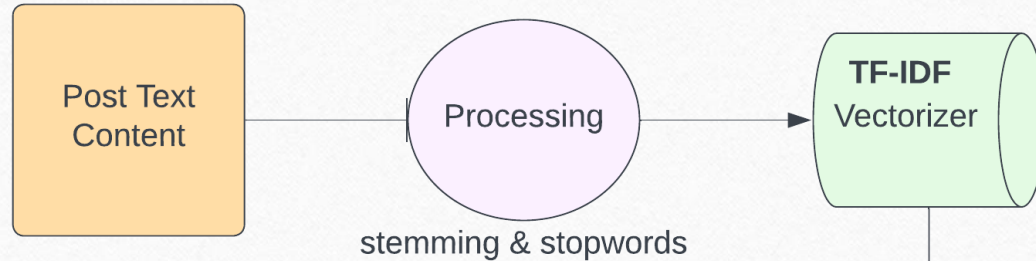
TF-IDF Matrix

```
array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
         0.        ],
        [0.        , 0.        , 0.10303694, ..., 0.        , 0.        ,
         0.        ],
        [0.        , 0.03807011, 0.        , ..., 0.        , 0.        ,
         0.        ],
        ...,
        [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
         0.        ],
        [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
         0.        ],
        [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
         0.        ]])
```

These are extremely black boxed explanations of the ML models to know how they were really implemented check out the Jupyter notebooks here!
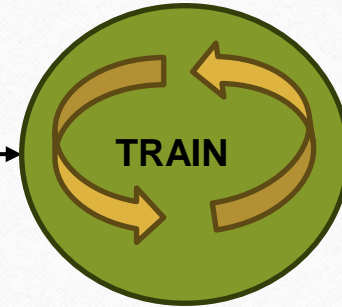
Our workflow(neural network define and train)

```
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(y_train.shape[1], activation='sigmoid')
])
```

Dense layer with 128 units
Dropout : to avoid overfitting
Final Dense layer → Num of target classes

TRAIN

10 EPOCHS

These are extremely black boxed explanations of the ML models to
know how they were really implemented check out the Jupyter notebooks here!

Evaluation metrics

```
          precision    recall  f1-score   support

       0       0.84      1.00      0.91       338
       1       0.82      1.00      0.90       329
       2       0.65      0.22      0.33       148
       3       0.58      0.47      0.52       190
       4       0.83      0.22      0.34        88
       5       1.00      0.07      0.13        42
       6       0.84      0.45      0.58       154
       7       0.00      0.00      0.00        47
       8       0.90      0.12      0.20        78

   micro avg       0.79      0.63      0.70      1414
   macro avg       0.72      0.39      0.44      1414
weighted avg       0.76      0.63      0.63      1414
 samples avg       0.79      0.67      0.69      1414
```



```python
# Calculate the number of correct predictions
correct_predictions = sum(y_test == y_pred)

# Calculate the total number of predictions
total_predictions = len(y_pred)

# Calculate the accuracy
accuracy = correct_predictions / total_predictions

# Print the accuracy
print("Accuracy:", accuracy)
```
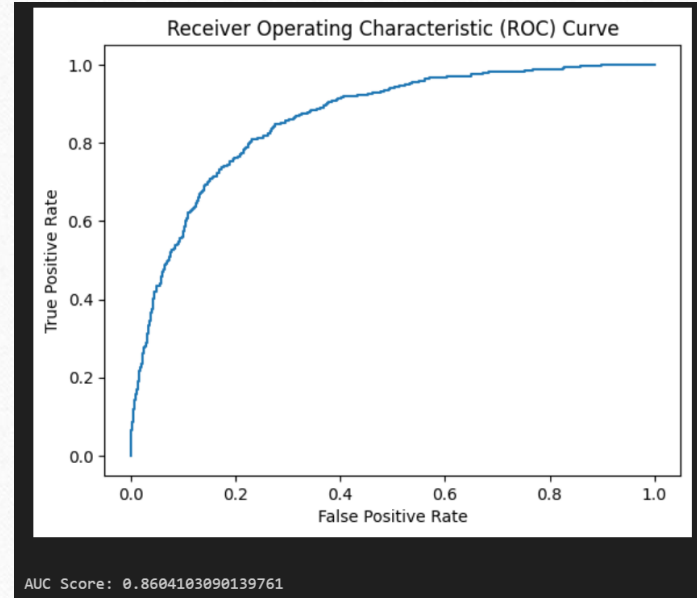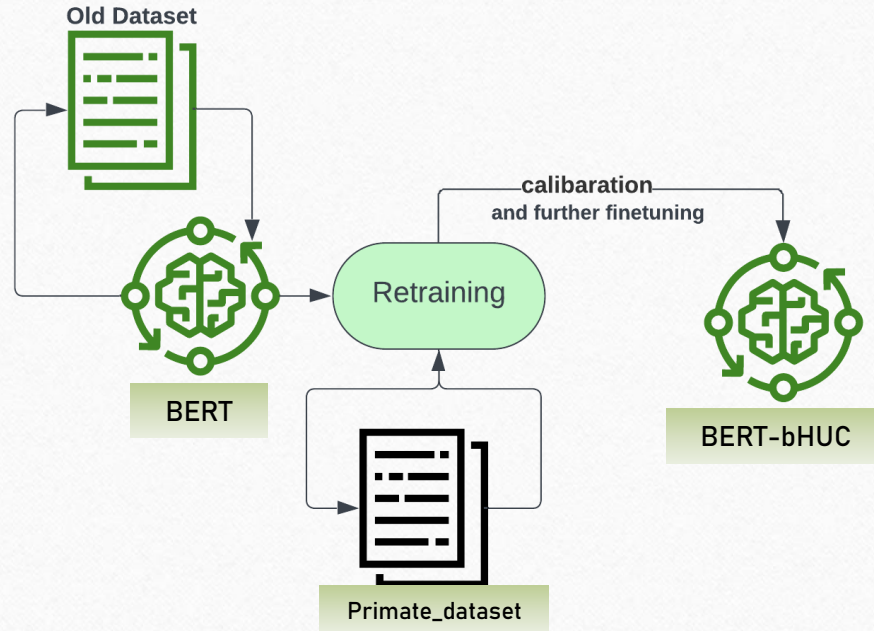
```
AUC Score: 0.8604103090139761
```

```
Accuracy: [0.80597015 0.82587065 0.66666667 0.58706468 0.76119403 0.91044776
 0.78109453 0.90049751 0.80597015]
```

Accuracy of each label

Our Second Approach(Transfer learning)

Old Dataset

calibaration and further finetuning

Retraining

BERT

BERT–bHUC

Primate_dataset

These are extremely black boxed explanations of the ML models to know how they were really implemented check out the Jupyter notebooks here!

Evaluation metrics


```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      1.00      0.91       338
           1       0.82      1.00      0.90       329
           2       0.53      0.62      0.57       148
           3       0.51      0.86      0.64       190
           4       0.43      0.48      0.45        88
           5       0.71      0.24      0.36        42
           6       0.58      0.72      0.64       154
           7       0.57      0.09      0.15        47
           8       0.60      0.42      0.50        78

   micro avg       0.68      0.79      0.73      1414
   macro avg       0.62      0.60      0.57      1414
weighted avg       0.68      0.79      0.72      1414
 samples avg       0.68      0.81      0.71      1414

Overall Accuracy: 10.47
```
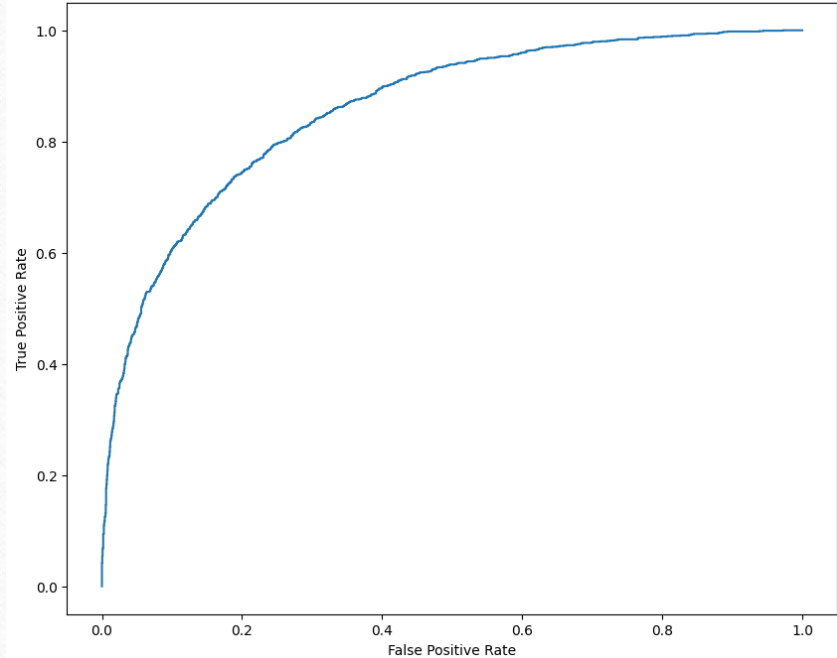

Receiver Operating Characteristic (ROC) Curve

As accuracy is not a great metric in multi-label classification with class imbalance we decided to choose a OneVsRest Classifier model as it performs better on other metrics

The Jupyter notebooks here!