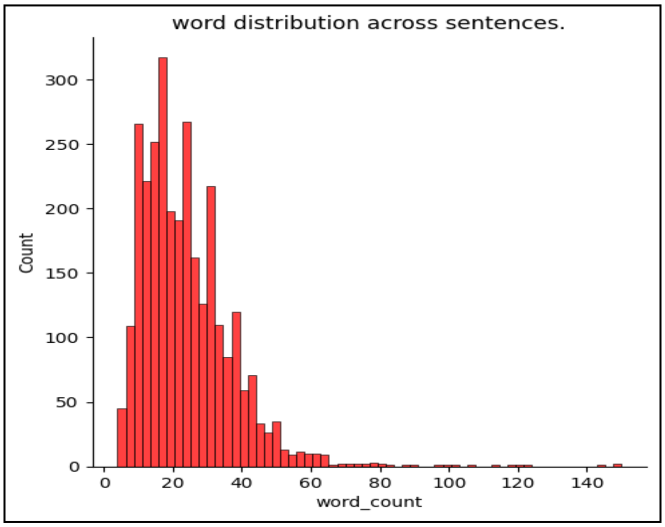
Project Description:

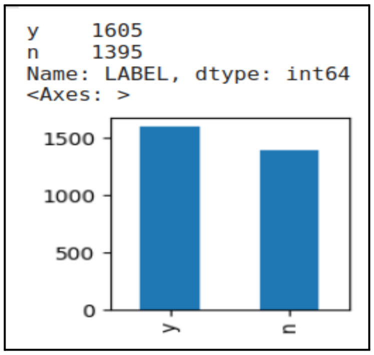
The objective of the project is to train, and evaluate a deep learning based classification model that has the capability to determine whether a particular relationship triplet, consisting of a subject, relation, and object, is expressed in a given sentence. To achieve this objective, the project will utilize the data columns, sentences, subject, predicates, and objects to make predictions regarding the value of the corresponding LABEL column.

Step 1: Data Exploration

1. Data consists of 3000 rows and 27 columns.
2. There are no null values in the data.
3. Below is the distribution of word count in each sentence of the dataset, We can see that most of the sentences have 10-60 words.

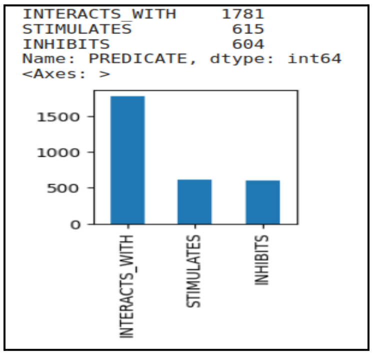


4. Below is the plot of count of categories of our LABEL column. We can conclude that there is no considerable imbalance in the data.



```
y    1605
n    1395
Name: LABEL, dtype: int64
<Axes: >
```

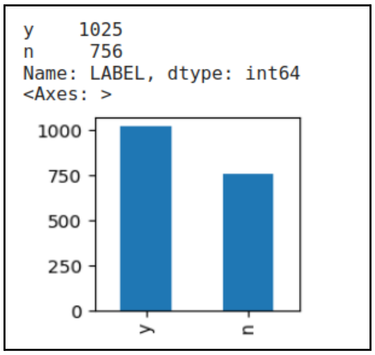
5. Analyzing the PREDICATE column distribution in the dataset. We can see the most frequently occurring relationship in the dataset is the "INTERACTS_WITH".



```
INTERACTS_WITH     1781
STIMULATES          615
INHIBITS            604
Name: PREDICATE, dtype: int64
<Axes: >
```

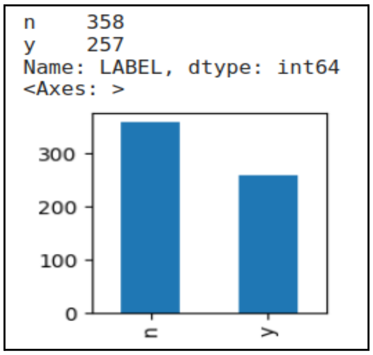
6. Label Distribution with respect to each value in PREDICATE column:

INTERACTS_WITH



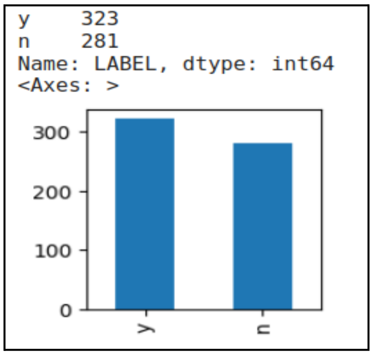
```
y    1025
n     756
Name: LABEL, dtype: int64
<Axes: >
```

INHIBITS



```
n    358
y    257
Name: LABEL, dtype: int64
<Axes: >
```

STIMULATES



```
y    323
n    281
Name: LABEL, dtype: int64
<Axes: >
```

Step 2: Modeling Techniques

1. BASELINE MODEL:

- To train the baseline model, I have used sentences, subjects, objects, and predicates columns from the dataset to create input features, and label columns as the target variable.
- I have utilized additional special tags to encode the subject and object entities in the sentences itself, using <e1>SUBJECT</e1> and <e2>OBJECT</e2>, respectively, to enable the use of these entities as input features and passed Predicate as additional feature to Bert Tokenizer for generating token embedding.

Input Data example:

Original Sentence: Nor did administration of SA, diflunisal or ASA itself impair the anti-aggregatory effect of a fresh test dose of ASA.

Sentence after Encoding: Nor did administration of **<e1>**SA**</e1>**, diflunisal or **<e2>**ASA**</e2>** itself impair the anti-aggregatory effect of a fresh test dose of ASA

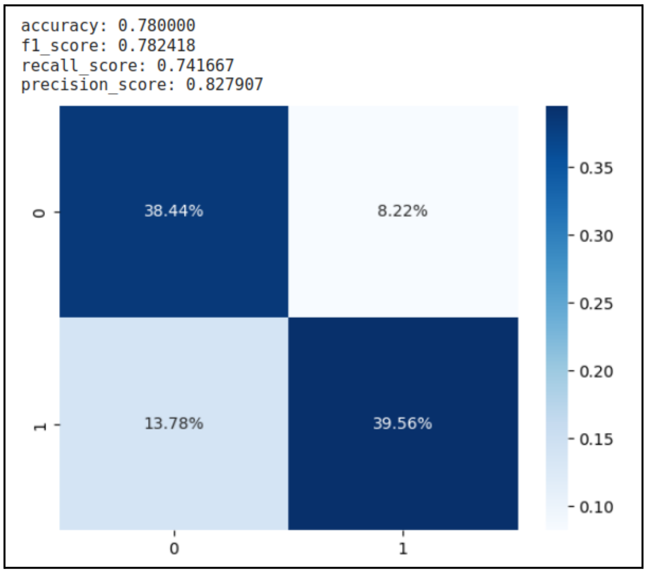
Final Input: Nor did administration of **<e1>**SA**</e1>**, diflunisal or **<e2>**ASA**</e2>** itself impair the anti-aggregatory effect of a fresh test dose of ASA [SEP] INHIBITS_WITH

I split the dataset into train, eval, and test datasets. The training set consists of 2100 rows, while the test set and eval set contain 450 rows each.

Model:

I have fine-tuned Bert-Model for classification using AutoModelForSequenceClassification architecture, trainer function, and used pretrained microsoft/BiomedNLP-PubMedBERT-base-uncased-abstract model. The reason for utilizing this pre-trained model is its state-of-the-art performance on several biomedical NLP tasks reported by several research papers.

Evaluation: This model has achieved an F1-score of **0.78** on the test dataset.



```
accuracy: 0.780000
f1_score: 0.782418
recall_score: 0.741667
precision_score: 0.827907
```

P.S: I have tried multiple combinations of batch size before reaching this result but I have removed that from the file to lower the size.

2. SETFIT MODEL AS A FEW SHOT APPROACH.

Recently I came across a very interesting paper Tunstall, Lewis, et al. "Efficient Few-Shot Learning Without Prompt" (https://arxiv.org/abs/2209.11055) which fine-tunes a Sentence Transformer model on a small number of labeled example followed by training a classifier head on the embeddings generated from the fine-tuned Sentence Transformer.

As I obtained a favorable outcome with my baseline model, I decided to take advantage of this technique to attempt and improve the model performance.

Similar to the baseline model, I have utilized sentences, subjects, objects, and predicates to generate input features, while the label serves as the target variable. However, I have made adjustments to the data preparation process for this model. Specifically, we transformed the subject, object, and predicate into a question format and appended it to the original sentence.
The rationale behind this approach is to incorporate the relationship triplet as a feature into the sentence itself, but directly adding the subject, predicate, and object would have implied that the relationship is true. Instead, I aimed to confound the model by adding it as a question.

Input Data example:

Original Sentence: Nor did administration of SA, diflunisal or ASA itself impair the anti-aggregatory effect of a fresh test dose of ASA.

Sentence after Encoding: Nor did administration of SA, diflunisal or ASA itself impair the anti-aggregatory effect of a fresh test dose of ASA. **Does SA INTERACTS_WITH ASA?**

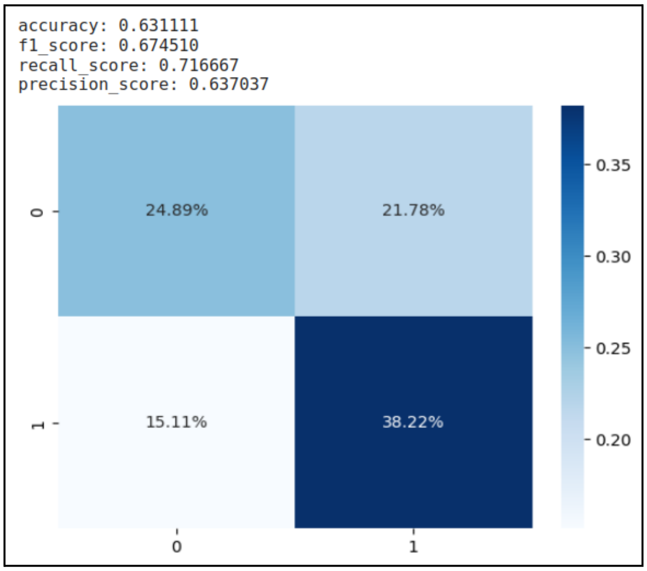
Final Input : The final input to the model was "Sentence_question" and "Target_Variable".

In order to fine-tune the setfit model, I employed the setfit trainer function provided by Hugging Face. The setfit model carries out contrastive learning on sentence embeddings during the initial fine-tuning stage, followed by utilizing the fine-tuned embeddings to train the scikit-learn head for classification. To accomplish this, I used the all-roberta-large pre-trained model as suggested by the paper "**sentence-transformers/all-roberta-large-v1**".

I have tried combination of few shot but adding only top 3:

1. **Few Shot with 20 examples for each of the six combinations of Predicate & Label.**
   The dataset for this model comprises a total of 120 sentences.I partitioned the dataset into three subsets, namely train, eval, and test datasets. The training set consists of 84 rows, while the test set and eval set contain 18 rows each.

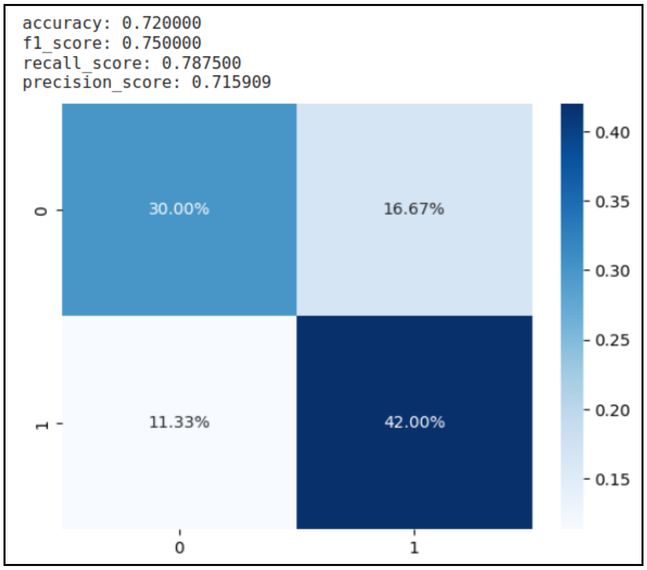
   Evaluation: Considering the small dataset, the model demonstrated satisfactory performance with an F1-score of 0.67.



```
accuracy: 0.631111
f1_score: 0.674510
recall_score: 0.716667
precision_score: 0.637037
```

2. **Few Shot with 150 examples for each of the six combinations of Predicate & Label.**
   The dataset for this model comprises a total of 900 sentences.I partitioned the dataset into three subsets, namely train, eval, and test datasets. The training set consists of 630 rows, while the test set and eval set contain 135 rows each.

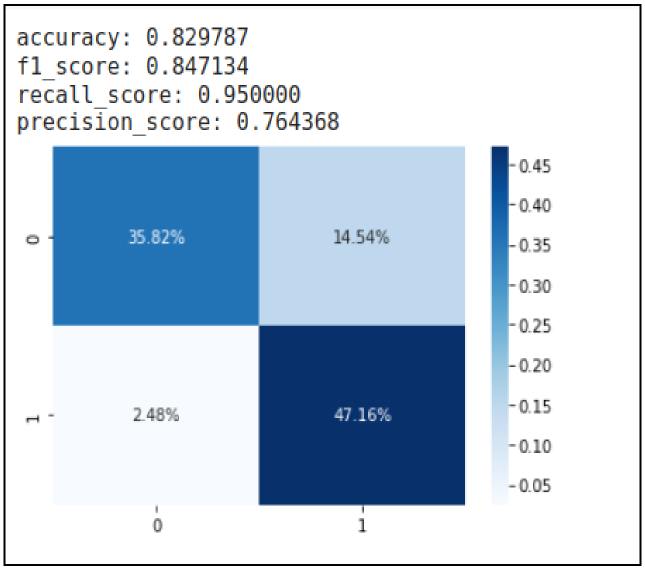
   Evaluation: By simply increasing the number of sentences in the model input to 630, the model was able to achieve a similar outcome to the baseline model, with an F1-score of around 0.75.



```
accuracy: 0.720000
f1_score: 0.750000
recall_score: 0.787500
precision_score: 0.715909
```

3. **Fine-tuned Setfit Model using 50% dataset.**
   The dataset for this model comprises a total of 1889 sentences keeping equal distribution for all three PREDICATE. I partitioned the dataset into three subsets, namely train, eval, and test datasets. The training set consists of 1315 rows, while the test set and eval set contain 287 rows each.

   Evaluation: This model surpassed all the previous models achieving and overall accuracy of 0.84.



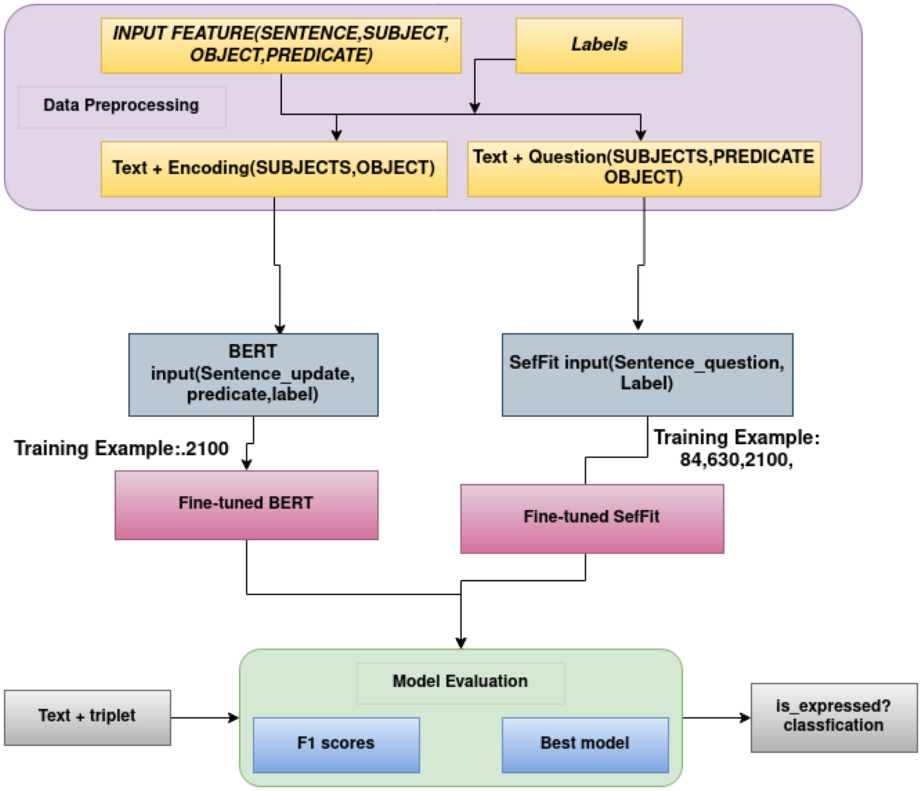
```
accuracy: 0.829787
f1_score: 0.847134
recall_score: 0.950000
precision_score: 0.764368
```

P.S: I ran this particular experiment in a separate notebook just to optimize time, hence not part of the submission notebook.

One thing noticeable is that baseline models have better precision compared to all Few-shot models, whereas recall is better in Few-Shot models.

Flow Chart:

Below is the flowchart of both attempted approaches.


INPUT FEATURE(SENTENCE,SUBJECT, OBJECT,PREDICATE)
Labels
Data Preprocessing
Text + Encoding(SUBJECTS,OBJECT)
Text + Question(SUBJECTS,PREDICATE OBJECT)
BERT input(Sentence_update, predicate,label)
SefFit input(Sentence_question, Label)
Training Example:.2100
Training Example: 84,630,2100,
Fine-tuned BERT
Fine-tuned SefFit
Model Evaluation
Text + triplet
F1 scores
Best model
is_expressed? classfication

Conclusion:

I really enjoyed working on this problem, It was a good dataset to work with. If I had more time, I would have attempted to solve the given task as a question-answering problem. Additionally, I would have spent more time in hyperparameter tuning for the models.