# IIT KGP Machine Learning Hackathon

## Introduction

There were two tasks in the given problem statement, **Task1** was to use any framework we are comfortable with for the Research Paper Publishability Assessment. In this we had to determine which all research papers can be published and which all cannot. It can be assessed based on very many factors like grammatical wise, sentence structuring and very many more. **Task2** was  to select the conference where the research paper could be published and for this we had to mandatory use the Pathway framework.

### Approach for Task1

For Task 1 first of all we took all the publishable and non publishable dataset which were given to us and converted it into a dataframe , after this we actually paraphrased the text. After paraphrasing we had got around 177 text with labels '1' and '0' wherein 1 represents published and 0 represents non-published. This dataset was then divided into train and test and the splitting was done using `Stratified Shuffle Split (sss)`. The sss actually takes the dataset in such a manner that the labelled classes are taken in almost equal ratio. After that we had added a pipeline and used a ModerBert model it failed to connect because of the new upgrades in the pytorch library so we decided to move on with the `MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7(MLD)`. As this model contained all the prerequisite files in the existing pytorch model and no more further updates or external library delivery was required we decided to move on with the `MLD` model. After using the classifier model we went ahead with the preprocessed data and implemented the Stratified Shuffle Split on K-fold model wherein k was given to be 3. Using the K-fold we had figured out the F1 score for 5 epochs were from the second epoch onwards we were getting F1 score to be 1.

After more in depth analysis of the approach and various cross-verifications we came to the conclusion that the model was not giving accurate results.

Thus we went ahead with a second approach which was BiLSTM RNN. The BiLSTM was used so that the text that we are giving to the system to learn would change its weight and loss taking consideration of the previous text that the system had went through. Like if the the system goes through on research paper which is published then to another which one is not published then the loss that it would be incorporating would be too high and the system would learn like what is the reason that there is so much of high loss coming. This would also be true when the system would be going from one published research paper to another research paper which is published, here also loss would be incurred but not too much as was in the case of non-publishable research paper.

Here also each and every research paper was broken down into chunks and each chunk contained 512 words.

Using all the 15 research paper we had trained the model and the loss was also giving satisfying values. But when it was tested on an unknown dataset the accuracy was coming out to be very low. The reason we found out to be the lack of dataset collection. The dataset was too small for the implementation of the BiLSTM model.

After failing in one more model we decided to go on with the LinearSVC Classifier.As mentioned earlier that we had quite less amount of dataset thus we switched, the Linear SVC the multi-dimensional data is separated by a hyperplane. Therefore the kernel function helps in finding the SVC in higher dimensions and giving us a relationship between the data in higher dimensions.

The LinearSVC classifier model contained the nltk library which is used to remove the stopwords. After removing the stopwords the sentences were then passed through TF-IDF Vectorization. The text in the transform part of TF-IDF was taken into account. Along with these the Label column from the dataset was also taken into account.

Both these columns were used for training and testing purposes with the test size as 20%. On this training and testing data the LinearSVC was implemented and the predictions were obtained. In this the Precision came out to be 0.5 and F1 score to be 0.6667 .

The result was added into a csv file named “Prediction.csv” which contained columns named PDF file, Predictions.

RESULTS:

```
print("\nEvaluation Summary:")
print(results)
```

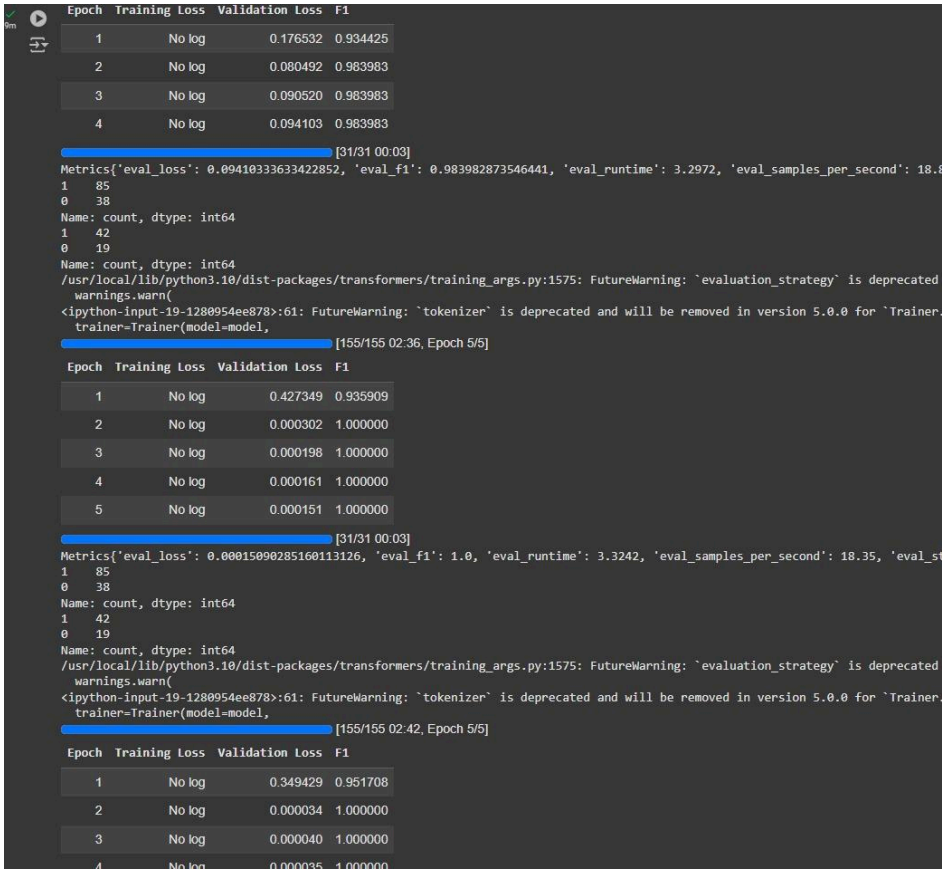
```
Classification Report:
                precision    recall  f1-score   support

Non-Publishable       0.00      0.00      0.00         1
    Publishable       0.50      1.00      0.67         1

       accuracy                           0.50         2
      macro avg       0.25      0.50      0.33         2
   weighted avg       0.25      0.50      0.33         2


Evaluation Summary:
      Metric     Value
0  Precision  0.500000
1     Recall  1.000000
2   F1-Score  0.666667
```

Publication and Non Publication papers using SVC.



| Epoch | Training Loss | Validation Loss | F1 |
|---|---|---|---|
| 1 | No log | 0.176532 | 0.934425 |
| 2 | No log | 0.080492 | 0.983983 |
| 3 | No log | 0.090520 | 0.983983 |
| 4 | No log | 0.094103 | 0.983983 |

[31/31 00:03]

```
Metrics{'eval_loss': 0.09410333633422852, 'eval_f1': 0.983982873546441, 'eval_runtime': 3.2972, 'eval_samples_per_second': 18.8
1    85
0    38
Name: count, dtype: int64
1    42
0    19
Name: count, dtype: int64
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1575: FutureWarning: `evaluation_strategy` is deprecated
  warnings.warn(
<ipython-input-19-1280954ee878>:61: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.
  trainer=Trainer(model=model,
```

[155/155 02:36, Epoch 5/5]

| Epoch | Training Loss | Validation Loss | F1 |
|---|---|---|---|
| 1 | No log | 0.427349 | 0.935909 |
| 2 | No log | 0.000302 | 1.000000 |
| 3 | No log | 0.000198 | 1.000000 |
| 4 | No log | 0.000161 | 1.000000 |
| 5 | No log | 0.000151 | 1.000000 |

[31/31 00:03]

```
Metrics{'eval_loss': 0.0001509028516011 3126, 'eval_f1': 1.0, 'eval_runtime': 3.3242, 'eval_samples_per_second': 18.35, 'eval_s
1    85
0    38
Name: count, dtype: int64
1    42
0    19
Name: count, dtype: int64
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1575: FutureWarning: `evaluation_strategy` is deprecated
  warnings.warn(
<ipython-input-19-1280954ee878>:61: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.
  trainer=Trainer(model=model,
```

[155/155 02:42, Epoch 5/5]

| Epoch | Training Loss | Validation Loss | F1 |
|---|---|---|---|
| 1 | No log | 0.349429 | 0.951708 |
| 2 | No log | 0.000034 | 1.000000 |
| 3 | No log | 0.000040 | 1.000000 |
| 4 | No log | 0.000035 | 1.000000 |

The F1 score by using MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7 model.

```
Token indices sequence length is longer than the specified maximum sequence length

Predictions:
     PDF File       Prediction
0    P035.pdf  Non-Publishable
1    P047.pdf  Non-Publishable
2    P093.pdf  Non-Publishable
3    P090.pdf  Non-Publishable
4    P057.pdf  Non-Publishable
..        ...              ...
130  P042.pdf  Non-Publishable
131  P038.pdf  Non-Publishable
132  P101.pdf  Non-Publishable
133  P040.pdf  Non-Publishable
134  P024.pdf  Non-Publishable

[135 rows x 2 columns]

Value Counts:
Prediction
Non-Publishable    135
Name: count, dtype: int64
```

Results by using MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7. As all the results are coming out to be 135 Non-Publishable we didn't use it.

## Approach for Task2

The starting process was through the use of the prediction CSV file which we created for classifying the papers into publishable and non publishable using SVC with "linear" kernel and parameter set for max features as 5000, and then from which the data were filtered only for the papers to be published. The dataset provided included 135 reference papers to download from a link on the shared drive which was given in the PS. From the zip file of the dataset, bash commands were applied in appending research papers into a list called data to process effectively.

An SVC model was developed with a "linear" kernel and balanced weights. The model here is mostly the same from the one used in Task 1. We also include the NLTK library to remove stopwords. The cleaned sentences were then vectorized using the TF-IDF method, with emphasis on the transformed text. The label column from the dataset was also used for classification.

A dataframe publishable_paper was generated, which contains only the publishable papers found in the prediction column. Based on this, a new dataframe named new_df was created and iterated for all text and corresponding PDF file names as columns from the dataframe publishable_paper. Finally, the text column in new_df was saved into another dataframe named final_df.

Preprocessing techniques were applied to the text column of final_df, and the processed text was saved in a new column called processed_text. Then, the existing TF-IDF vectorizer was applied to the processed_text, and predictions were generated for the reference papers.

Lastly, a new dataframe was created, containing the following columns:

PDF File Name: The name of the PDF file.

Prediction: Whether the paper is publishable or not.

Conference: The conference at which the paper is likely to be submitted.

This streamlined approach allowed for the efficient prediction and classification of the reference papers.

RESULTS:

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| CVPR | 1.00 | 1.00 | 1.00 | 1 |
| EMNLP | 0.50 | 1.00 | 0.67 | 1 |
| KDD | 0.00 | 0.00 | 0.00 | 1 |
| NeurIPS | 0.50 | 1.00 | 0.67 | 1 |
| TMLR | 0.00 | 0.00 | 0.00 | 1 |
| accuracy | | | 0.60 | 5 |
| macro avg | 0.40 | 0.60 | 0.47 | 5 |
| weighted avg | 0.40 | 0.60 | 0.47 | 5 |

Evaluation Summary:

| | Metric | Value |
|---|---|---|
| 0 | Precision | 0.400000 |
| 1 | Recall | 0.600000 |
| 2 | F1-Score | 0.466667 |

Classification Report for the five conferences using the Linear SVC model

We got a 0.00 F1 score for the KDD and TMLR conference which is a second round off result. When we increase the precision limit we get a small non zero value.

RAG Architecture:

First of all we had used the google drive connector.It was implemented using the airbyte-ci connectors.The URL which was shared with us in the problem statement, we used that for connecting to the google drive but for some reasons we faced errors so had to switch to gdown command to download the research paper. Even if there has been an update like addition of research paper we would still be able to retrieve those papers for fitting our model.

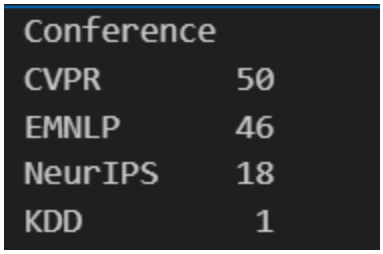
We had initialized the Pathway Vectorstore in which we stored the chunks of the text of each of the research papers by embedding them using the NLTK, TF-IDF Vectorizer.

Subsequently when we receive the chunk text of any other research paper that needs to be classified, cosine similarity search is performed and we get the text which has the maximum score.

We feed that text to the GROQ API along with the prediction of the conference which we got from our SVC model. What it does is provides us with the rationale that explains why this conference paper was predicted.

**Conclusion:**

Using the Linear SVC model we got 115 Publishable research papers and 20 non-publishable research papers for the non labelled dataset which was provided in the drive, which have been provided in the csv file with the paper ids. Later on using the same model we got 50 CVPR, 46 EMNLP, 18 NeurIPS and 1 KDD.



| Conference | |
|---|---|
| CVPR | 50 |
| EMNLP | 46 |
| NeurIPS | 18 |
| KDD | 1 |

Prediction for the conferences

We didn't get any result for the TMLR conference, the reason which we think is that it is a more generalized conference and the content of their research paper some-what coincides with that of other conferences. We had check the validation of that in OpenAI GPT and it clearly showed that the TMLR papers even got classified in other conferences.

**Future Insights:**

- If we had some more time we would have looked into properly integrating the pathway google drive connector in docker.

- The docker didn't support .ipynb file which was a major setback for us we had to convert everything to .py format which was time consuming for us but we could have brought better results.
- We could have ensemble various models for better classification results and might have got more accurate F1 scores like KNN, NAIVE BAYES, Decision Tree, XGBoost, etc...
- We would have also integrated a webpage.

**Appendix:**

1. aiohappyeyeballs==2.4.4
2. aiohttp==3.11.11
3. aiosignal==1.3.2
4. annotated-types==0.7.0
5. anyio==4.8.0
6. asttokens==3.0.0
7. attrs==24.3.0
8. beautifulsoup4==4.12.3
9. certifi==2024.12.14
10. charset-normalizer==3.4.1
11. click==8.1.8
12. colorama==0.4.6
13. comm==0.2.2
14. datasets==3.2.0
15. debugpy==1.8.11
16. decorator==5.1.1
17. dill==0.3.8
18. distro==1.9.0
19. docker==7.1.0
20. executing==2.1.0
21. filelock==3.16.1
22. frozenlist==1.5.0
23. fsspec==2024.9.0
24. gdown==5.2.0
25. groq==0.15.0
26. h11==0.14.0
27. httpcore==1.0.7
28. httpx==0.28.1
29. huggingface-hub==0.27.1
30. idna==3.10
31. ipykernel==6.29.5
32. ipython==8.31.0
33. jedi==0.19.2
34. Jinja2==3.1.5

35. joblib==1.4.2
36. jupyter_client==8.6.3
37. jupyter_core==5.7.2
38. MarkupSafe==3.0.2
39. matplotlib-inline==0.1.7
40. mpmath==1.3.0
41. multidict==6.1.0
42. multiprocess==0.70.16
43. nest-asyncio==1.6.0
44. networkx==3.4.2
45. nltk==3.9.1
46. numpy==2.2.1
47. packaging==24.2
48. pandas==2.2.3
49. parso==0.8.4
50. pathway==0.post1
51. pillow==11.1.0
52. platformdirs==4.3.6
53. prompt_toolkit==3.0.48
54. propcache==0.2.1
55. psutil==6.1.1
56. pure_eval==0.2.3
57. pyarrow==18.1.0
58. pydantic==2.10.5
59. pydantic_core==2.27.2
60. Pygments==2.19.1
61. PyPDF2==3.0.1
62. PySocks==1.7.1
63. python-dateutil==2.9.0.post0
64. pytz==2024.2
65. pywin32==308
66. PyYAML==6.0.2
67. pyzmq==26.2.0
68. regex==2024.11.6
69. requests==2.32.3
70. safetensors==0.5.2
71. scikit-learn==1.6.1
72. scipy==1.15.1
73. sentence-transformers==3.3.1
74. six==1.17.0
75. sniffio==1.3.1
76. soupsieve==2.6
77. stack-data==0.6.3

78. sympy==1.13.1
79. threadpoolctl==3.5.0
80. tokenizers==0.21.0
81. torch==2.5.1
82. tornado==6.4.2
83. tqdm==4.67.1
84. traitlets==5.14.3
85. transformers==4.48.0
86. typing_extensions==4.12.2
87. tzdata==2024.2
88. urllib3==2.3.0
89. wcwidth==0.2.13
90. xxhash==3.5.0
91. yarl==1.18.3