PIVONY

Topic: MultiLevelTextClassification Report

Submitted To:
Emre Calisir

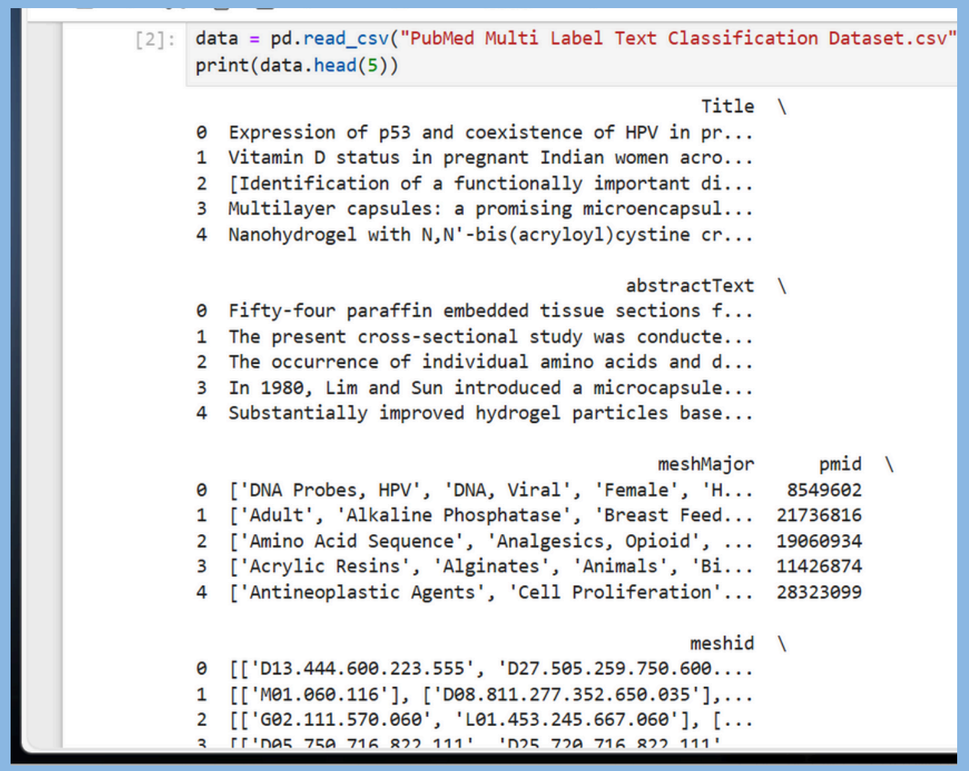
Submitted By:
Vishnu Prakash

Dataset

MultiLabelText Classification

CSV file Overview

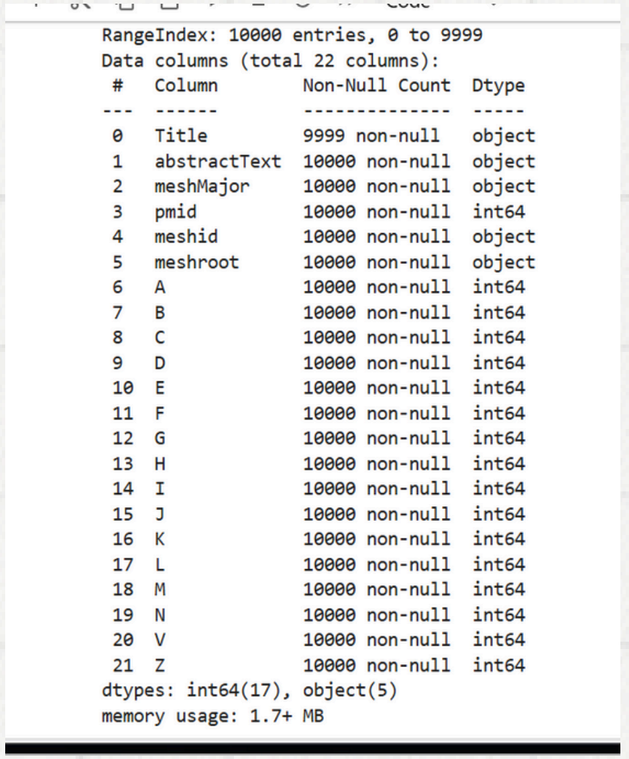
- The dataset is used for multi-label text classification tasks.
- It contains text samples and corresponding labels for classification.
- Each row represents a text instance with multiple associated categories.



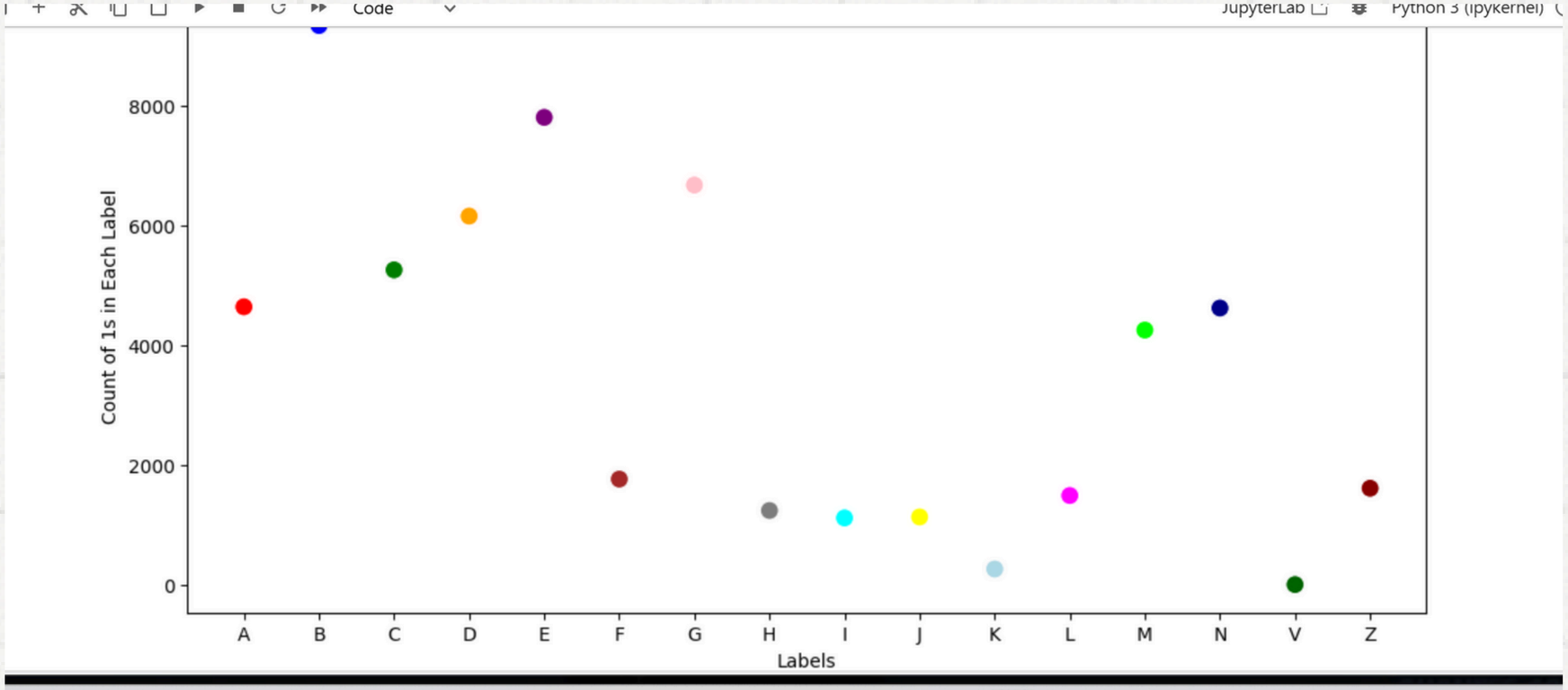
```
[2]: data = pd.read_csv("PubMed Multi Label Text Classification Dataset.csv"
     print(data.head(5))

                                                Title  \
0  Expression of p53 and coexistence of HPV in pr...
1  Vitamin D status in pregnant Indian women acro...
2  [Identification of a functionally important di...
3  Multilayer capsules: a promising microencapsul...
4  Nanohydrogel with N,N'-bis(acryloyl)cystine cr...

                                            abstractText  \
0  Fifty-four paraffin embedded tissue sections f...
1  The present cross-sectional study was conducte...
2  The occurrence of individual amino acids and d...
3  In 1980, Lim and Sun introduced a microcapsule...
4  Substantially improved hydrogel particles base...

                                           meshMajor      pmid  \
0  ['DNA Probes, HPV', 'DNA, Viral', 'Female', 'H...   8549602
1  ['Adult', 'Alkaline Phosphatase', 'Breast Feed...  21736816
2  ['Amino Acid Sequence', 'Analgesics, Opioid', ...  19060934
3  ['Acrylic Resins', 'Alginates', 'Animals', 'Bi...  11426874
4  ['Antineoplastic Agents', 'Cell Proliferation'...  28323099

                                              meshid  \
0  [['D13.444.600.223.555', 'D27.505.259.750.600....
1  [['M01.060.116'], ['D08.811.277.352.650.035'],...
2  [['G02.111.570.060', 'L01.453.245.667.060'], [...
3  [['D05.750.716.822.111', 'D25.720.716.822.111'...
```

Checking Null Values

The dataset was checked for any missing or null values, and none were found. This ensures that all data entries are complete, with no gaps in the columns. Having no null values means the dataset is ready for further analysis or model training without the need for handling missing data.



```
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 22 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Title         9999 non-null   object
 1   abstractText  10000 non-null  object
 2   meshMajor     10000 non-null  object
 3   pmid          10000 non-null  int64
 4   meshid        10000 non-null  object
 5   meshroot      10000 non-null  object
 6   A             10000 non-null  int64
 7   B             10000 non-null  int64
 8   C             10000 non-null  int64
 9   D             10000 non-null  int64
 10  E             10000 non-null  int64
 11  F             10000 non-null  int64
 12  G             10000 non-null  int64
 13  H             10000 non-null  int64
 14  I             10000 non-null  int64
 15  J             10000 non-null  int64
 16  K             10000 non-null  int64
 17  L             10000 non-null  int64
 18  M             10000 non-null  int64
 19  N             10000 non-null  int64
 20  V             10000 non-null  int64
 21  Z             10000 non-null  int64
dtypes: int64(17), object(5)
memory usage: 1.7+ MB
```

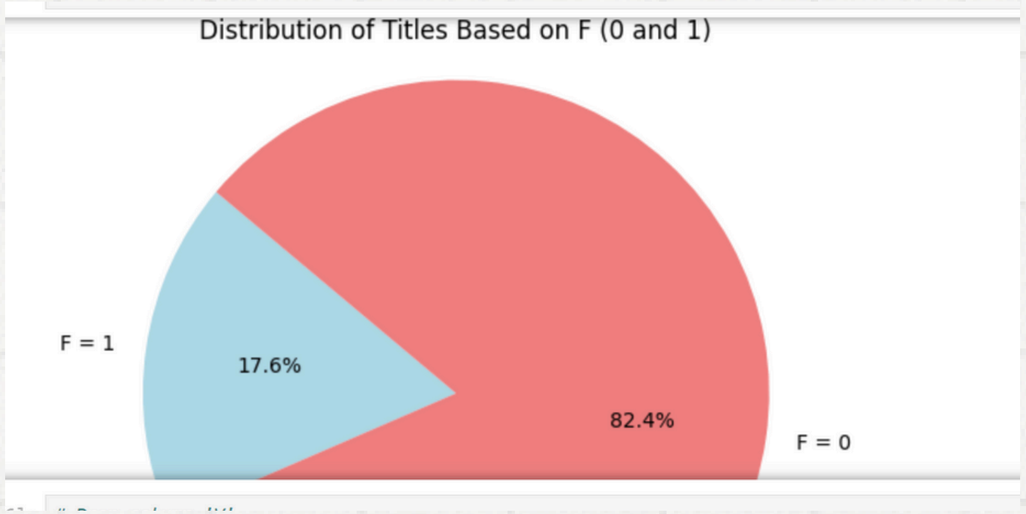


Graph Representation of Column Counts:

The graph above displays the count of values for each column, labeled from A to Z. Each bar in the graph corresponds to a specific column, showing how many entries are present. This helps in understanding the distribution of data across the columns and identifying any imbalances or trends in the dataset.

Pie Chart Representation of Column "F":

The pie chart illustrates the percentage of values 1 and 0 present in column "F." It shows the proportion of data where the value is either 1 or 0, making it easy to visualize the distribution. This helps in understanding how balanced the data is within column "F."



Program Overview:

The program shown in the image counts how many rows in columns A, B, C, ..., Z have all values equal to either 1 or 0. For example, if in the first row, column A has a value of 0 and all other columns (B, C, ..., Z) in that row also have 0, the count increases by 1. The same applies for rows where all columns have a value of 1. If the values don't match across all columns, the count is not increased. This helps in identifying how many rows are fully consistent with either all 1s or all 0s.

[6]:
```
# Drop column 'V'
data = data.drop(columns=['V'])
```

[7]:
```
# Define the columns to check
columns = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'Z']

# Create a mask for rows where all specified columns are 1
all_ones_mask = (data[columns] == 1).all(axis=1)
all_zeros_mask = (data[columns] == 0).all(axis=1)

# Count the rows where all columns are 1
count_all_ones = all_ones_mask.sum()

# Count the rows where all columns are 0
count_all_zeros = all_zeros_mask.sum()

# Print the results
print(f"Count of rows where all columns are 1: {count_all_ones}")
print(f"Count of rows where all columns are 0: {count_all_zeros}")
```

```
Count of rows where all columns are 1: 0
Count of rows where all columns are 0: 81
```

Row Removal and Output:

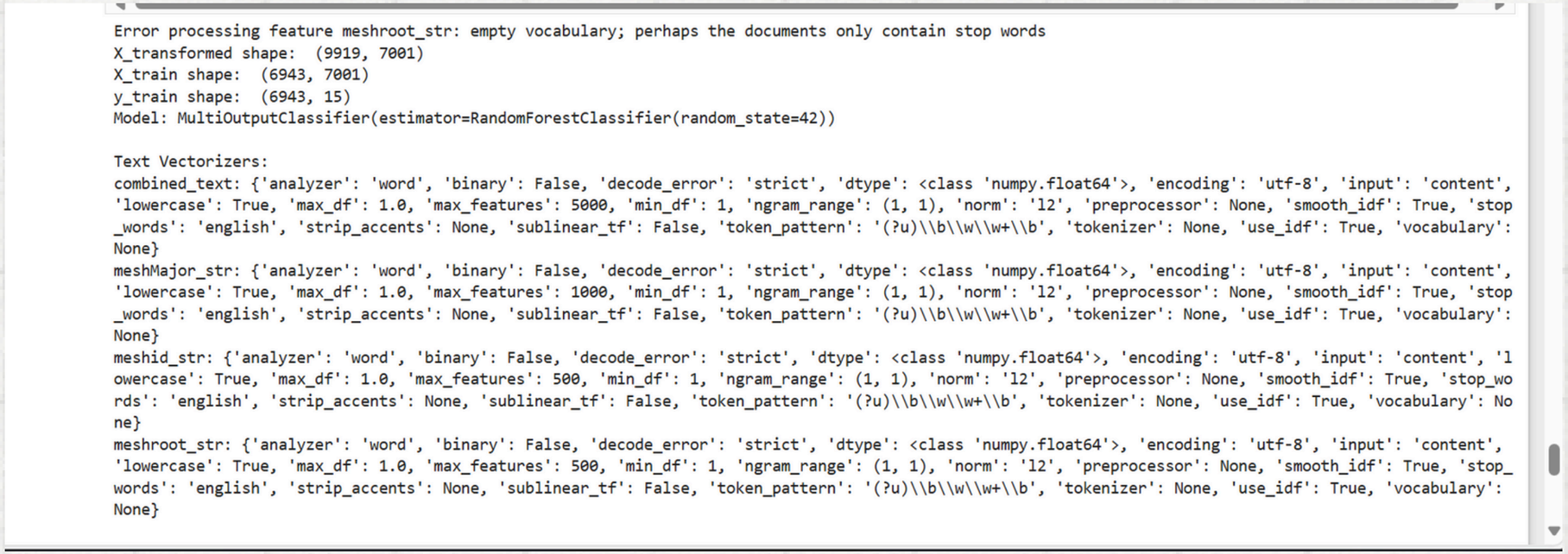
All rows where the values in every column were equal to 0 have been removed. The output now shows that no rows have all columns equal to 0, ensuring cleaner data.

[9]:
```
# Drop the rows where all specified columns are 0
data = data[~all_zeros_mask]
```

[10]:
```
# Define the columns to check
columns = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'Z']

# Create a mask for rows where all specified columns are 1
all_ones_mask = (data[columns] == 1).all(axis=1)
all_zeros_mask = (data[columns] == 0).all(axis=1)



# Count the rows where all columns are 1
count_all_ones = all_ones_mask.sum()

# Count the rows where all columns are 0
count_all_zeros = all_zeros_mask.sum()

# Print the results
print(f"Count of rows where all columns are 1: {count_all_ones}")
print(f"Count of rows where all columns are 0: {count_all_zeros}")
```

```
Count of rows where all columns are 1: 0
Count of rows where all columns are 0: 0
```

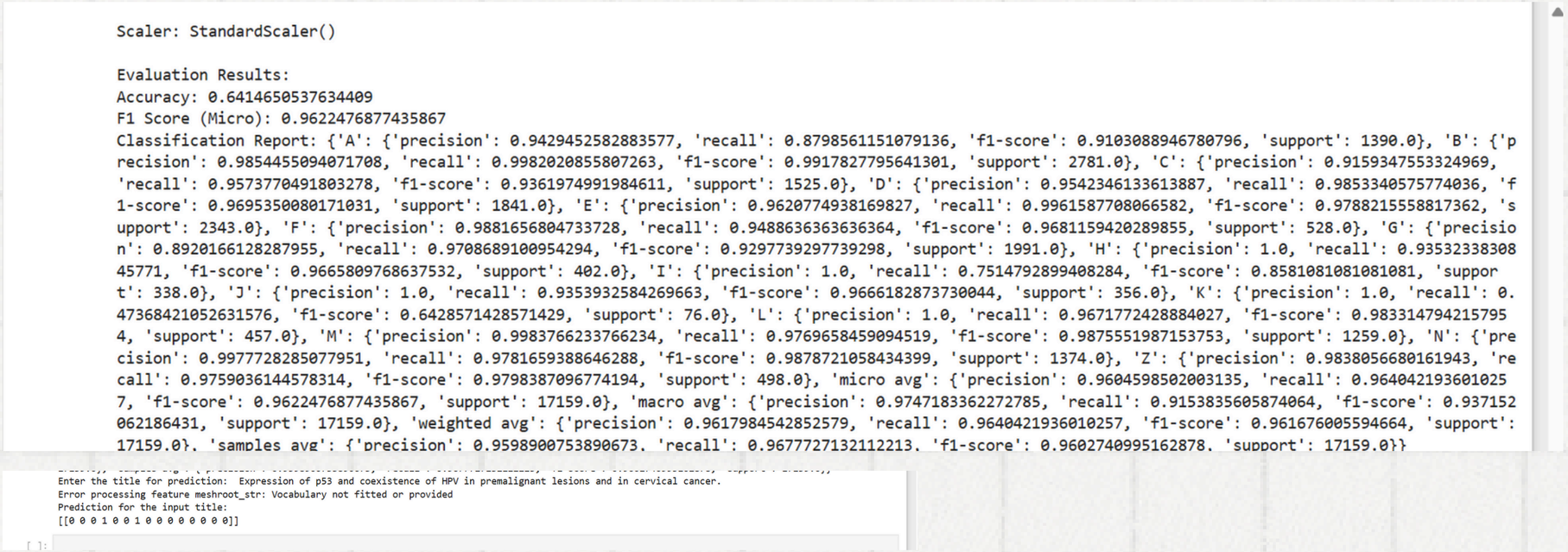
Model and Text Vectorizers Overview:



```
Error processing feature meshroot_str: empty vocabulary; perhaps the documents only contain stop words
X_transformed shape:  (9919, 7001)
X_train shape:  (6943, 7001)
y_train shape:  (6943, 15)
Model: MultiOutputClassifier(estimator=RandomForestClassifier(random_state=42))

Text Vectorizers:
combined_text: {'analyzer': 'word', 'binary': False, 'decode_error': 'strict', 'dtype': <class 'numpy.float64'>, 'encoding': 'utf-8', 'input': 'content',
'lowercase': True, 'max_df': 1.0, 'max_features': 5000, 'min_df': 1, 'ngram_range': (1, 1), 'norm': 'l2', 'preprocessor': None, 'smooth_idf': True, 'stop
_words': 'english', 'strip_accents': None, 'sublinear_tf': False, 'token_pattern': '(?u)\\b\\w\\w+\\b', 'tokenizer': None, 'use_idf': True, 'vocabulary':
None}
meshMajor_str: {'analyzer': 'word', 'binary': False, 'decode_error': 'strict', 'dtype': <class 'numpy.float64'>, 'encoding': 'utf-8', 'input': 'content',
'lowercase': True, 'max_df': 1.0, 'max_features': 1000, 'min_df': 1, 'ngram_range': (1, 1), 'norm': 'l2', 'preprocessor': None, 'smooth_idf': True, 'stop
_words': 'english', 'strip_accents': None, 'sublinear_tf': False, 'token_pattern': '(?u)\\b\\w\\w+\\b', 'tokenizer': None, 'use_idf': True, 'vocabulary':
None}
meshid_str: {'analyzer': 'word', 'binary': False, 'decode_error': 'strict', 'dtype': <class 'numpy.float64'>, 'encoding': 'utf-8', 'input': 'content', 'l
owercase': True, 'max_df': 1.0, 'max_features': 500, 'min_df': 1, 'ngram_range': (1, 1), 'norm': 'l2', 'preprocessor': None, 'smooth_idf': True, 'stop_wo
rds': 'english', 'strip_accents': None, 'sublinear_tf': False, 'token_pattern': '(?u)\\b\\w\\w+\\b', 'tokenizer': None, 'use_idf': True, 'vocabulary': No
ne}
meshroot_str: {'analyzer': 'word', 'binary': False, 'decode_error': 'strict', 'dtype': <class 'numpy.float64'>, 'encoding': 'utf-8', 'input': 'content',
'lowercase': True, 'max_df': 1.0, 'max_features': 500, 'min_df': 1, 'ngram_range': (1, 1), 'norm': 'l2', 'preprocessor': None, 'smooth_idf': True, 'stop_
words': 'english', 'strip_accents': None, 'sublinear_tf': False, 'token_pattern': '(?u)\\b\\w\\w+\\b', 'tokenizer': None, 'use_idf': True, 'vocabulary':
None}
```

The image shows the model and Text Vectorizers during the running phase. For the model, I used MultiOutputClassifier with RandomForestClassifier because it handles labeled data effectively. Since the CSV file contains labeled data, RandomForestClassifier was chosen for its ability to classify multi-label datasets efficiently.

Scaler, Evaluation Results, and User Input Prediction:



```
Scaler: StandardScaler()

Evaluation Results:
Accuracy: 0.6414650537634409
F1 Score (Micro): 0.9622476877435867
Classification Report: {'A': {'precision': 0.9429452582883577, 'recall': 0.8798561151079136, 'f1-score': 0.9103088946780796, 'support': 1390.0}, 'B': {'p
recision': 0.9854455094071708, 'recall': 0.9982020855807263, 'f1-score': 0.9917827795641301, 'support': 2781.0}, 'C': {'precision': 0.9159347553324969,
'recall': 0.9573770491803278, 'f1-score': 0.9361974991984611, 'support': 1525.0}, 'D': {'precision': 0.9542346133613887, 'recall': 0.9853340575774036, 'f
1-score': 0.9695350080171031, 'support': 1841.0}, 'E': {'precision': 0.9620774938169827, 'recall': 0.9961587708066582, 'f1-score': 0.9788215558817362, 's
upport': 2343.0}, 'F': {'precision': 0.9881656804733728, 'recall': 0.9488636363636364, 'f1-score': 0.9681159420289855, 'support': 528.0}, 'G': {'precisio
n': 0.8920166128287955, 'recall': 0.9708689100954294, 'f1-score': 0.9297739297739298, 'support': 1991.0}, 'H': {'precision': 1.0, 'recall': 0.93532338308
45771, 'f1-score': 0.9665809768637532, 'support': 402.0}, 'I': {'precision': 1.0, 'recall': 0.7514792899408284, 'f1-score': 0.8581081081081081, 'suppor
t': 338.0}, 'J': {'precision': 1.0, 'recall': 0.9353932584269663, 'f1-score': 0.9666182873730044, 'support': 356.0}, 'K': {'precision': 1.0, 'recall': 0.
47368421052631576, 'f1-score': 0.6428571428571429, 'support': 76.0}, 'L': {'precision': 1.0, 'recall': 0.9671772428884027, 'f1-score': 0.983314794215795
4, 'support': 457.0}, 'M': {'precision': 0.9983766233766234, 'recall': 0.9769658459094519, 'f1-score': 0.9875551987153753, 'support': 1259.0}, 'N': {'pre
cision': 0.9977728285077951, 'recall': 0.9781659388646288, 'f1-score': 0.9878721058434399, 'support': 1374.0}, 'Z': {'precision': 0.9838056680161943, 're
call': 0.9759036144578314, 'f1-score': 0.9798387096774194, 'support': 498.0}, 'micro avg': {'precision': 0.9604598502003135, 'recall': 0.964042193601025
7, 'f1-score': 0.9622476877435867, 'support': 17159.0}, 'macro avg': {'precision': 0.9747183362272785, 'recall': 0.9153835605874064, 'f1-score': 0.937152
062186431, 'support': 17159.0}, 'weighted avg': {'precision': 0.9617984542852579, 'recall': 0.9640421936010257, 'f1-score': 0.961676005594664, 'support':
17159.0}, 'samples avg': {'precision': 0.9598900753890673, 'recall': 0.9677727132112213, 'f1-score': 0.9602740995162878, 'support': 17159.0}}
Enter the title for prediction:  Expression of p53 and coexistence of HPV in premalignant lesions and in cervical cancer.
Error processing feature meshroot_str: Vocabulary not fitted or provided
Prediction for the input title:
[[0 0 0 1 0 0 1 0 0 0 0 0 0 0 0]]
```

The image displays the data Scaler used for preprocessing, along with the evaluation results that assess the model's performance. Additionally, we can see the prediction results based on user-provided input. This shows how the model processes and predicts outcomes for new data after scaling and evaluation.

Final reflections and future steps

We can use this model for political purposes, such as analyzing the speech of a leader and gathering public opinion. For example, we can provide a sentence for voting, where people can express their opinion by liking it (+1 for 1) or disliking it (+1 for 0). This way, we can calculate which sentences are most effective on the public.

Thank you!