**[Video link - NLP final project part-B](https://drive.google.com/file/d/1plL4whRaykE4tR2gPSsBdObBVV3rcXZf/view?usp=sharing)**

# Part B : News Article Classification

REPORT

## Step-1: environment

Loaded all the libraries and stopwords

## Step-2: loading the dataset

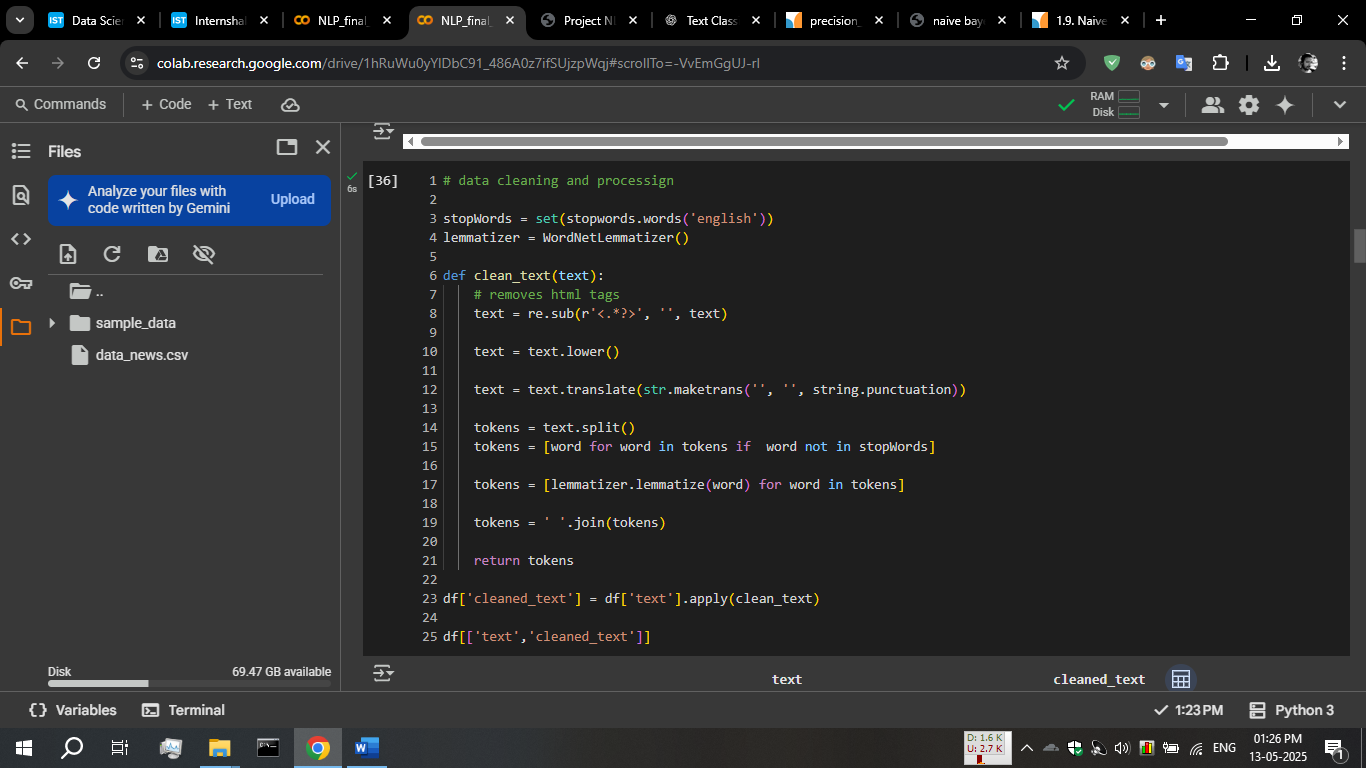
I used colab upload option to load the data ,

and then read\_csv function of pandas to reaed the file content.

## Step-3: test pre-processing

First I removed the rows containg null values using dropna function

df = df.dropna(axis=0)



our dataset containing many text columns, so I combined them into one

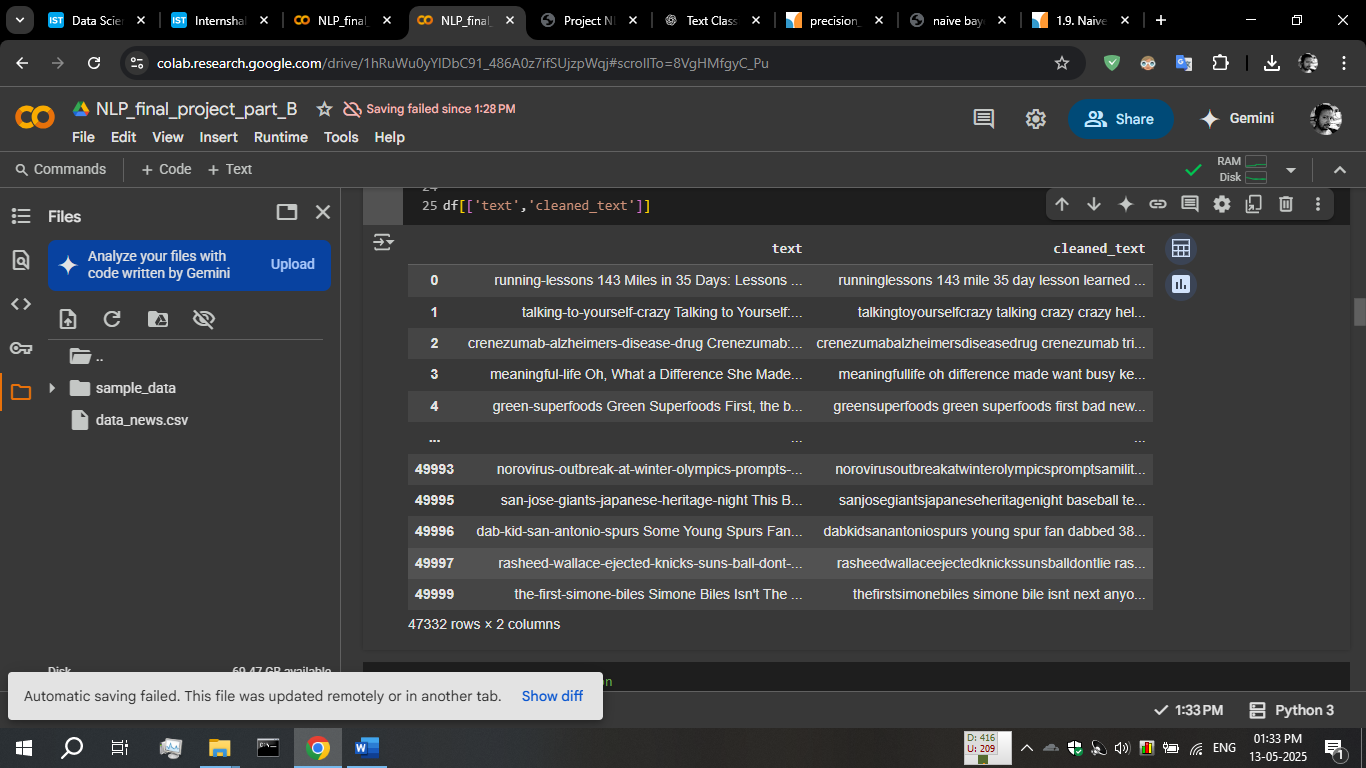
df['text'] = df['keywords'] + ' ' +df['headline'] + ' ' + df['short\_description']

then for cleaning

I used regular expressions to remove html tags, then maketrans and translate function to remove any punctuation and stop words

After it we apply lemmatization

Data after cleaning in cleaned\_text column



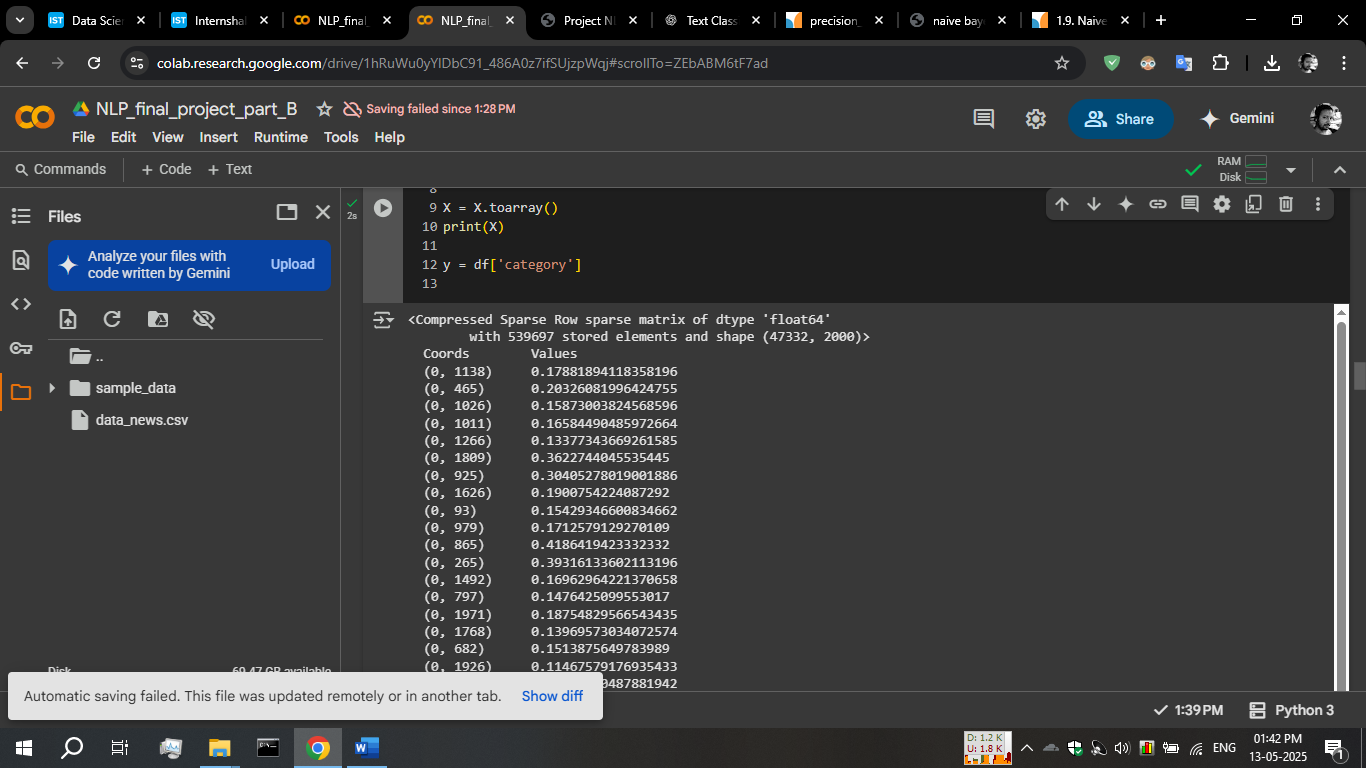
## Step 4: feature extraction and vectorization

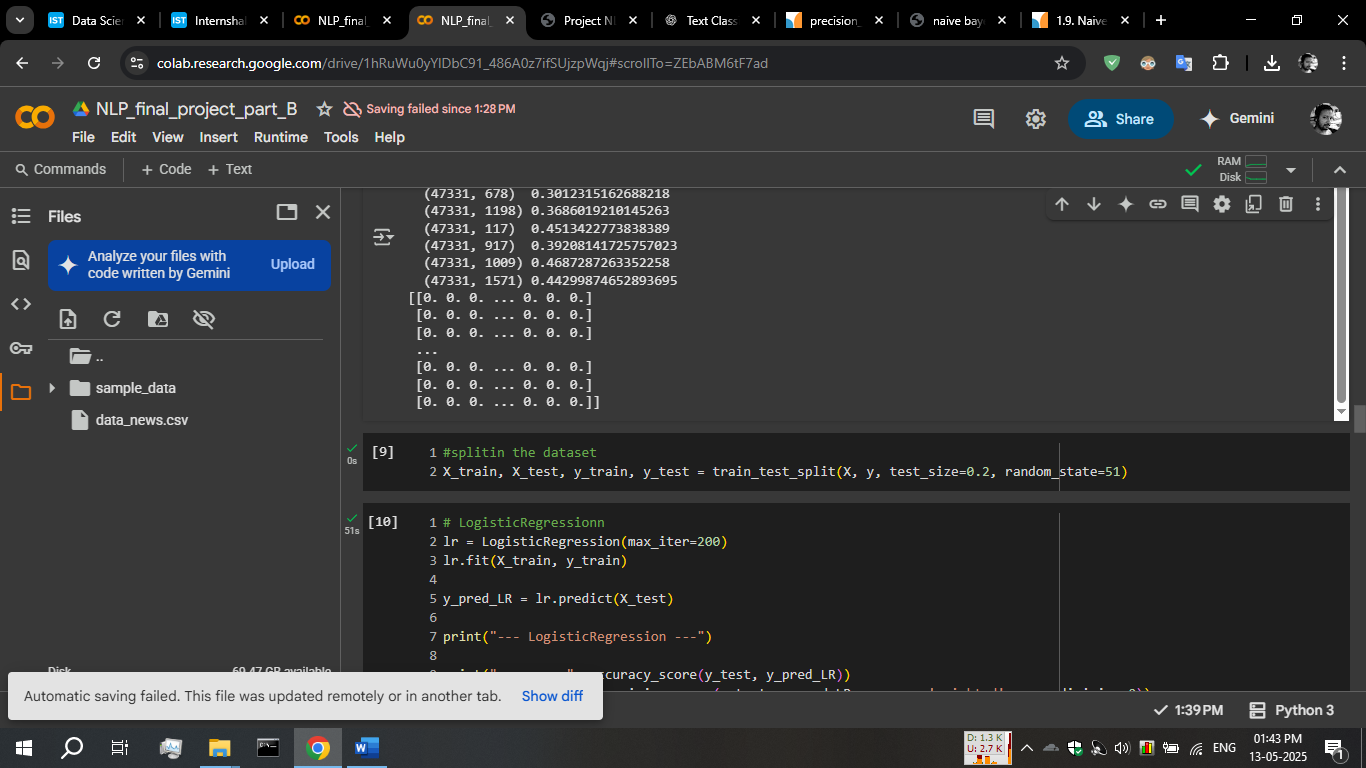
I used TfidfVectorizer class from sklearn.feature\_extraction.text,

tfidf = TfidfVectorizer(max\_features=2000)

X = tfidf.fit\_transform(df['cleaned\_text'])

print(X)

 outputs this 🡪

After toarray(),

## Step -4 : training the model

I use LogisticRegression, naive bayes – MultinomialNB , SVC-linear model for training

# LogisticRegressionn

lr = LogisticRegression(max\_iter=200)

lr.fit(X\_train, y\_train)

y\_pred\_LR = lr.predict(X\_test)

# naïve bayes - MultinomialNB - it is used for word classification

nb = MultinomialNB()

nb.fit(X\_train, y\_train)

y\_pred\_NB = nb.predict(X\_test)

# LinearSVC

svc = LinearSVC()

svc.fit(X\_train, y\_train)

y\_pred\_SVC = svc.predict(X\_test)

I trained all models with same training and test data

## Step-5: evaluating the models

All model preforming well, logisticRegression is better in all of them

--- LogisticRegression ---

accuracy: 0.8163092848843351

precision: 0.817225583855853

recall: 0.8163092848843351

f1 score: 0.816550059427474

report:

precision recall f1-score support

BUSINESS 0.80 0.78 0.79 901

ENTERTAINMENT 0.80 0.80 0.80 956

FOOD & DRINK 0.87 0.88 0.87 967

PARENTING 0.79 0.78 0.79 899

POLITICS 0.77 0.75 0.76 906

SPORTS 0.90 0.89 0.90 989

STYLE & BEAUTY 0.90 0.86 0.88 953

TRAVEL 0.79 0.79 0.79 974

WELLNESS 0.74 0.80 0.77 944

WORLD NEWS 0.82 0.81 0.81 978

accuracy 0.82 9467

macro avg 0.82 0.82 0.82 9467

weighted avg 0.82 0.82 0.82 9467

--- naive bayes ---

accuracy: 0.8025773740361255

precision: 0.80599239191142

recall: 0.8025773740361255

f1 score: 0.8032511411590265

report:

precision recall f1-score support

BUSINESS 0.80 0.73 0.77 901

ENTERTAINMENT 0.82 0.78 0.80 956

FOOD & DRINK 0.84 0.88 0.86 967

PARENTING 0.70 0.77 0.73 899

POLITICS 0.80 0.75 0.78 906

SPORTS 0.91 0.86 0.88 989

STYLE & BEAUTY 0.90 0.81 0.85 953

TRAVEL 0.75 0.80 0.78 974

WELLNESS 0.73 0.80 0.76 944

WORLD NEWS 0.79 0.83 0.81 978

accuracy 0.80 9467

macro avg 0.80 0.80 0.80 9467

weighted avg 0.81 0.80 0.80 9467

--- LinearSVC ---

accuracy: 0.8155698743002007

precision: 0.8156090760570802

recall: 0.8155698743002007

f1 score: 0.8153909932571692

report:

precision recall f1-score support

BUSINESS 0.80 0.83 0.82 901

ENTERTAINMENT 0.81 0.79 0.80 956

FOOD & DRINK 0.85 0.86 0.86 967

PARENTING 0.77 0.78 0.77 899

POLITICS 0.79 0.76 0.77 906

SPORTS 0.89 0.93 0.91 989

STYLE & BEAUTY 0.89 0.85 0.87 953

TRAVEL 0.79 0.77 0.78 974

WELLNESS 0.76 0.78 0.77 944

WORLD NEWS 0.80 0.80 0.80 978

accuracy 0.82 9467

macro avg 0.81 0.81 0.81 9467

weighted avg 0.82 0.82 0.82 9467

Thank you