

Text As Data Coursework Report

Q1.a) The dataset that I have chosen is from Kaggle. It consists of 23,481 rows of news text which we will use for training our model along with subject for each row which will become the labels. There are a bunch of other columns available but for the sake of simplicity we will ignore them as they are not needed to train our models. There are 6 subjects namely: Government News, Middle East, News, US News, left-news, politics. I have selected this dataset as it met all the requirements for our model training and had plenty of training data available. Any news application that works with displaying news can use this automatic classification to segregate the news.

b) There are 6 labels as mentioned above:

- Government News
- Middle East
- News
- US News
- left-news
- politics

The text column of the dataset consists of a string which is the news. We will train our model on this text and predict the subjects which are the labels of each news article. No preprocessing was needed on this dataset as it was clean and the labels were already defined. There are 6 labels which falls nicely in our range of 3 to 10. The text column consisting of the news will be used for classification.

c) No, the dataset is not split into train/test/validate sets. It also exceeds our maximum row limit of 10,000. We will thus do random sampling in order to meet this requirement and then use 60/20/20 split to define our train, test and validation datasets.

Here are the label counts for the train dataset.

✓

0s

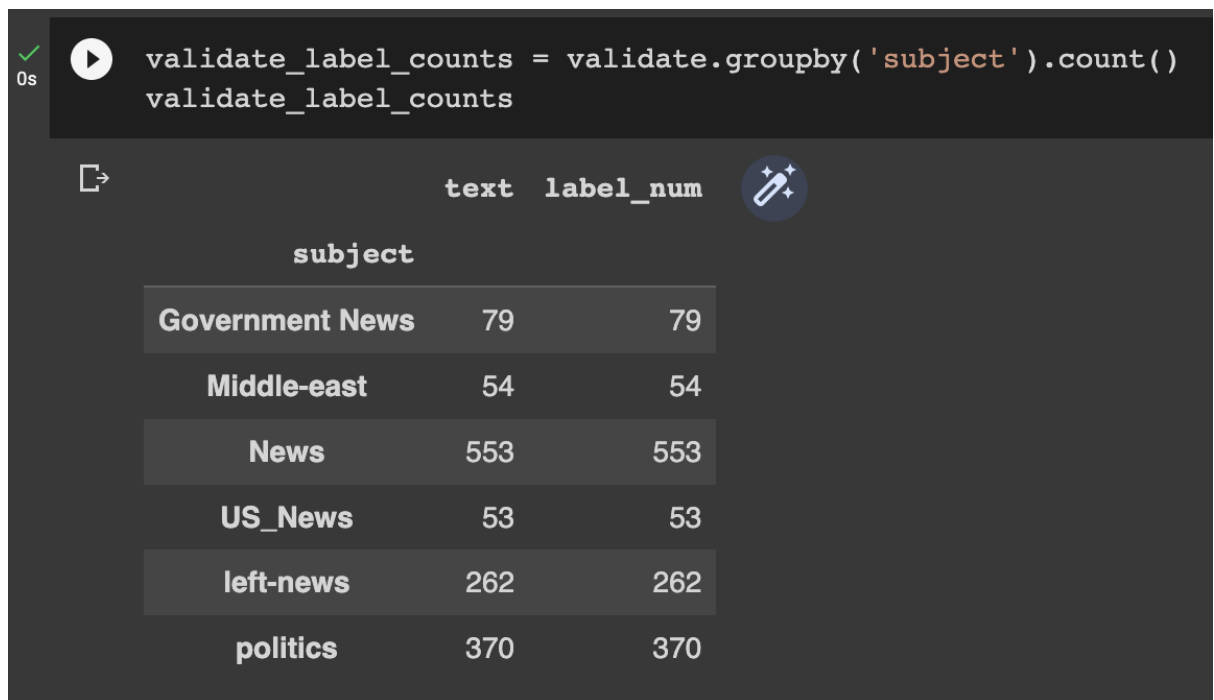
▶

```
train_label_counts = train.groupby('subject').count()
train_label_counts
```

	text	label_num
subject		
Government News	289	289
Middle-east	146	146
News	1633	1633
US_News	130	130
left-news	788	788
politics	1127	1127

We can see that the total number of labels in the training dataset comes out to be 4113. Therefore, there are 4113 rows in our training dataset. Upon some inspection, we see that the “news” label has the maximum number of texts(1633), followed by “politics” (1127), “left-news” (788), “Government News” (289), “Middle-east(“ 146) and “US_News” (130). We can say that the distribution of labels is somewhat balanced. This balanced distribution will ensure that our model is trained sufficiently on each label and is able to predict them confidently.

Next, we will look at the distribution across labels on the validate dataset.



In this case, we can see that the distribution is more imbalanced. The "News" category still has the largest number of text samples (553), but the other categories have relatively fewer samples: "politics" (370), "left-news" (262), "Government News" (79), "Middle-east" (54), and "US_News" (53).

This imbalance may affect our results as the validation dataset should have distributions somewhat similar to our training dataset.

Moving on to the testing dataset,

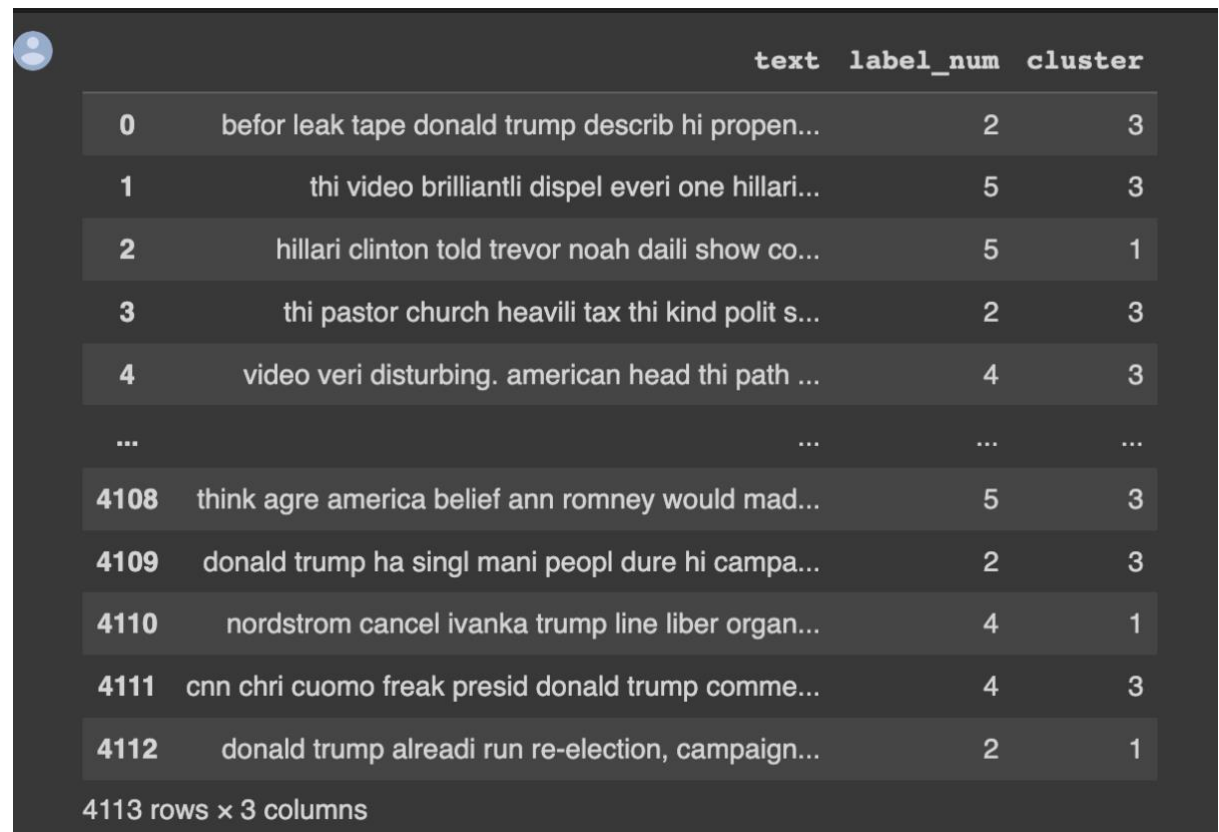


From the table we can see that the "News" category having the largest number of text samples (556), followed by "politics" (393), "left-news" (236), "Government News" (103), "Middle-east" (40), and "US_News" (44).

Q2.

a) A few examples of documents belonging to different cluster are:

In this iteration of the algorithm, clusters 1 and 3 were filled.



The screenshot shows a Jupyter Notebook interface with a table of document clustering results. The table has three columns: 'text', 'label_num', and 'cluster'. The rows are indexed from 0 to 4112. The table shows documents assigned to clusters 1, 3, and 4. The 'label_num' column contains values 2, 5, 5, 2, 4, ..., 5, 2, 4, 4, 2. The 'cluster' column contains values 3, 3, 1, 3, 3, ..., 3, 3, 1, 3, 1. The text column contains snippets of news articles related to Donald Trump and Hillary Clinton. At the bottom of the table, it says '4113 rows x 3 columns'.

	text	label_num	cluster
0	befor leak tape donald trump describ hi propen...	2	3
1	thi video brilliantli dispel everi one hillari...	5	3
2	hillari clinton told trevor noah daili show co...	5	1
3	thi pastor church heavili tax thi kind polit s...	2	3
4	video veri disturbing. american head thi path ...	4	3
...
4108	think agre america belief ann romney would mad...	5	3
4109	donald trump ha singl mani peopl dure hi campa...	2	3
4110	nordstrom cancel ivanka trump line liber organ...	4	1
4111	cnn chri cuomo freak presid donald trump comme...	4	3
4112	donald trump already run re-election, campaign...	2	1

4113 rows x 3 columns

Here is another iteration of the algorithm where only clusters 0 and 2 were filled. Here are the values in cluster 0.

	text	label_num	cluster
0	befor leak tape donald trump describ hi propen...	2	0
1	thi video brilliantli dispel everi one hillari...	5	0
2	hillari clinton told trevor noah daili show co...	5	0
3	thi pastor church heavili tax thi kind polit s...	2	0
4	video veri disturbing. american head thi path ...	4	0
...
4108	think agre america belief ann romney would mad...	5	0
4109	donald trump ha singl mani peopl dure hi campa...	2	0
4110	nordstrom cancel ivanka trump line liber organ...	4	0
4111	cnn chri cuomo freak presid donald trump comme...	4	0
4112	donald trump already run re-election, campaign...	2	0

4099 rows × 3 columns

Here are the values in cluster 2.

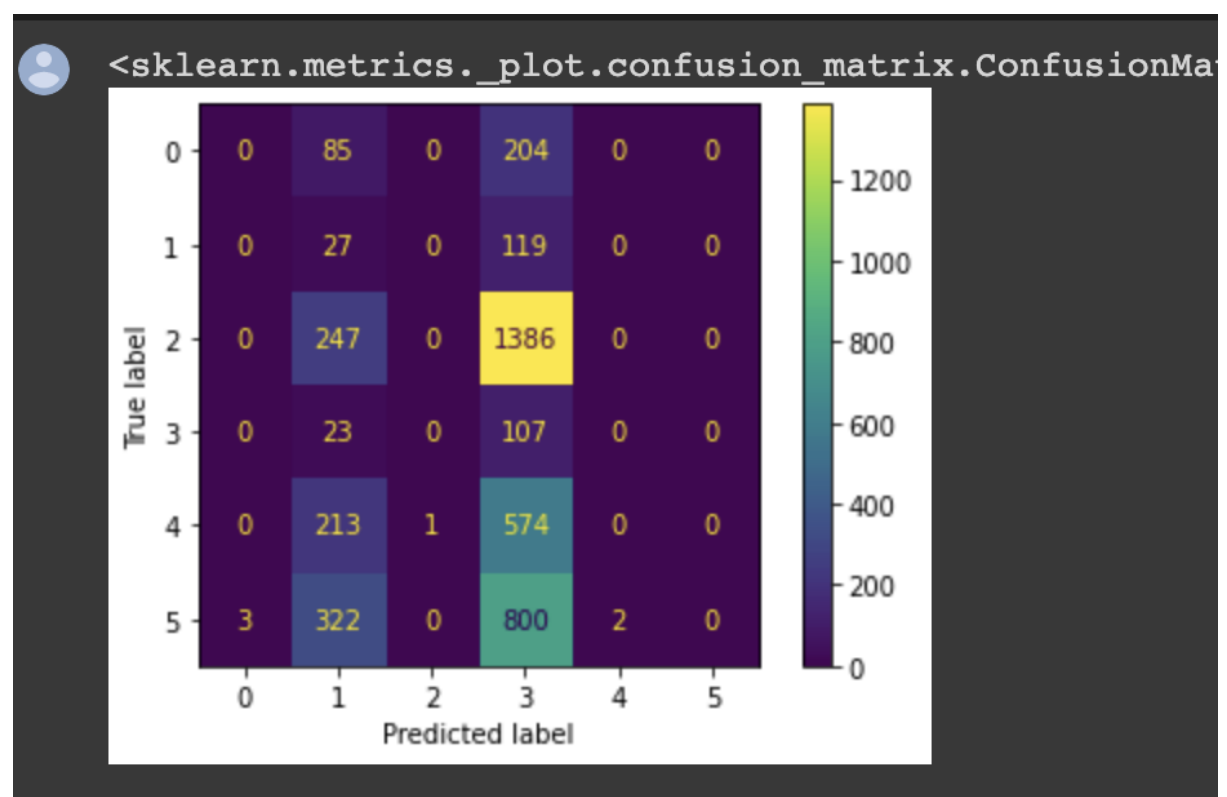
	text	label_num	cluster
150	boom!courtesi of: comicallyincorrect.com	4	2
262	https://www.youtube.com/watch?v=gqxwf-teya	5	2
404	https://www.youtube.com/watch?v=6vn1mabekik	5	2
495	https://www.youtube.com/watch?v=iioeiumawro	4	2
634	https://www.youtube.com/watch?v=8dsddbqf828	4	2
772	https://www.youtube.com/watch?v=8mehk5ewcza	5	2
854	https://www.youtube.com/watch?v=pjeoojypnck	4	2
2241	https://www.youtube.com/watch?v=ism-p8e-d7i	5	2
2614	https://www.youtube.com/watch?time_continue=13...	5	2
2954	https://www.youtube.com/watch?time_continue=2&...	0	2
3093	https://www.youtube.com/watch?v=yrxmfmgoptk	5	2
3130	https://www.youtube.com/watch?v=9lnyx_dwzza	4	2
3351	https://fedup.wpengine.com/wp-content/uploads/...	4	2
3574	https://fedup.wpengine.com/wp-content/uploads/...	5	2

By comparing the results of the above two iterations, we can say that:

- In the first iteration, clusters 1 and 3 were filled, but both of them had somewhat equal number of values and both had texts in them
- In the second iteration, clusters 0 and 2 were filled, but most values were in cluster 0 and cluster 2 only had those documents which had youtube links in them. Therefore, the clustering algorithm did manage to separate links from normal text.
- This suggests that clustering is only able to separate the documents into two labels even when there is a higher value of k. The original dataset has 6 labels and therefore clustering is not a good option.

b) No, the clusters do not make sense. This is assuming the fact that we were expecting the clusters to pick up one label each. However, maximum number of documents are shared by just two clusters (in the example case it is cluster 1 and cluster 3 or 0 and 2) with virtually no documents in the other three clusters. All the topics appear in both the clusters. However, since the k-means algorithm takes random centroids, it may not always be the case that clusters 1 and 3 are filled. But we can surely say that only 2 of the 5 clusters will be filled by the algorithm.

c) Here is a confusion matrix of k-means clustering on the dataset with k=5. (iteration where clusters 1 and 3 were filled)



The labels are as follows:

- 'Government News': 0,
- 'Middle-east': 1,

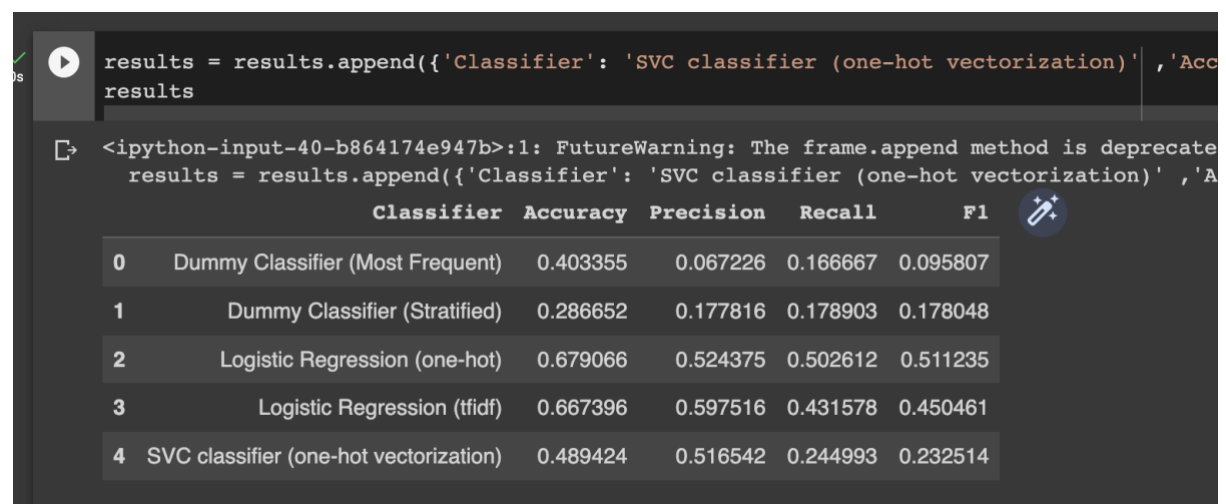
- 'News': 2,
- 'US_News': 3,
- 'left-news': 4,
- 'politics': 5

d) Looking at the confusion matrix, one can deduce the following (for the sake of simplicity, we will talk about our example case, but the same will be true for all instances of the algorithm, only the cluster numbers will change depending on the random initial centroids) :

- Cluster 1 and 3 share the majority of the documents.
- Cluster 3 has a very large number of documents belong to label 2 which is “news” label whereas cluster 1 has maximum documents belonging to label 5 which is “politics”. Although the documents in cluster 1 are somewhat evenly distributed.
- Cluster 3 has a large number of documents from all labels.
- Cluster 0 and cluster 4 only have documents from label 5.
- No cluster seems to be able to pick up a single label only. This means that k-means clustering with k=5 has not been successful on our dataset.

Q3.

a) Here is the table showing the evaluations of each classifier



The screenshot shows a Jupyter Notebook interface. At the top, there is a code cell with the following code: `results = results.append({'Classifier': 'SVC classifier (one-hot vectorization)', 'Accuracy': 0.232514, 'Precision': 0.244993, 'Recall': 0.516542, 'F1': 0.232514}, ignore_index=True)`. Below the code cell, there is a warning message: `<ipython-input-40-b864174e947b>:1: FutureWarning: The frame.append method is deprecated and will be removed in a future version. Use pandas.concat instead.`. Below the warning, there is a table with 5 columns: Classifier, Accuracy, Precision, Recall, and F1. The table contains 5 rows of data.

	Classifier	Accuracy	Precision	Recall	F1
0	Dummy Classifier (Most Frequent)	0.403355	0.067226	0.166667	0.095807
1	Dummy Classifier (Stratified)	0.286652	0.177816	0.178903	0.178048
2	Logistic Regression (one-hot)	0.679066	0.524375	0.502612	0.511235
3	Logistic Regression (tfidf)	0.667396	0.597516	0.431578	0.450461
4	SVC classifier (one-hot vectorization)	0.489424	0.516542	0.244993	0.232514

Looking at the table we can see that the baseline classifiers perform poorly and have low accuracy as well as precision, recall and f1 score. This tells us that they are not effective models for my dataset.

On the other hand, all three classifiers performed better than the baselines. In terms of accuracy, Logistic regression with one-hot encoding performed the best followed closely by logistic regression with tfidf encoding. SVC classifier performed the worst among the three classifier, this may be because of poor parameter selection.

We will talk about the classifiers in three aspects:

- 1) Appropriate model 'fit': All three models fit better than the baseline models on the dataset. However, if we take accuracy as our primary statistic, we can say that the logistic regression with one-hot encoding fits the dataset best as it has the highest accuracy. It also had the highest recall and f1 scores among the classifiers. On the other hand, logistic regression with tfidf vectorization had a slightly lower accuracy but a higher precision. The SVC classifier had the worst accuracy, precision, recall and f1 score among the classifiers.
- 2) Data Distribution: From the label distribution table, we can see that the "News" label has the highest number count in the training dataset, whereas "Middle-East" and "US_news " labels are less represented which may make it more difficult for classifiers to correctly identify them. Therefore, models that have lower scores may have had difficulties with these labels.
- 3) Classifier Models: The logistic regression models performed better than the SVC classifier with one hot vectorization. This may have been due to the fact that SVC is sensitive to parameters chosen and we did not tune it. Both LR models performed somewhat decent but the LR with one hot encoding performed slightly better than LR with tfidf vectorization in terms of accuracy, recall and f1 score whereas LR tfidf was better in terms of precision. This may be due to the fact that one-hot representation is a better representation for our dataset.

b) For my own classifier, I have chosen a Linear SVC with TFIDF vectorization. The results are as follows:

	Classifier	Accuracy	Precision	Recall	F1
0	Dummy Classifier (Most Frequent)	0.403355	0.067226	0.166667	0.095807
1	Dummy Classifier (Stratified)	0.272064	0.156919	0.156833	0.156784
2	Logistic Regression (one-hot)	0.679066	0.524375	0.502612	0.511235
3	Logistic Regression (tfidf)	0.667396	0.597516	0.431578	0.450461
4	SVC classifier (one-hot vectorization)	0.489424	0.516542	0.244993	0.232514
5	Linear SVC classifier (tfidf)	0.688549	0.554097	0.508969	0.522751

As you can see, a linear SVC with TFIDF gives us the highest accuracy, recall and f1 score. It only loses in precision to LR with tfidf vectorization. I chose this classifier as it works well with a wide range of text classification tasks and tfidf vectorization does good work to capture the importance of a word in the training data.

Q4.

Parameter tuning was performed on logistic regression model with tfidf vectorization. The results on the validation dataset were as follows:

	precision	recall	f1-score	support
Government News	0.42	0.20	0.27	79
Middle-east	0.51	0.41	0.45	54
News	0.95	1.00	0.98	553
US_News	0.50	0.47	0.49	53
left-news	0.46	0.39	0.42	262
politics	0.56	0.67	0.61	370
accuracy			0.70	1371
macro avg	0.57	0.52	0.54	1371
weighted avg	0.69	0.70	0.69	1371
Accuracy: 0.703				
Macro-averaged Precision: 0.569				
Macro-averaged Recall: 0.523				
Macro-averaged F1: 0.537				

We can see that there was an increase in performance across all metrics over the baseline LR TFIDF model. The most notable change was the increase in accuracy which went from 0.667 to 0.703. There was also a substantial increase in recall and f1 score.

The best parameters were calculated sequentially starting with sublinear_tf, followed by max_features, C value and finally solver. The best values came out to be:

- Sublinear_tf : True (tried values True and False)
- Max_features: 5000 (tried range None to 50k)
- C value : 10 (tried range 0.001 to 10000)
- Solver : liblinear

Q5.

a)

```
[ ] X_train = np.array([inp_features("[CLS]" + text, padding='max_length', max_length=1024, truncation=True)[0][0] for text in X_train])
X_test = np.array([inp_features("[CLS]" + text, padding='max_length', max_length=1024, truncation=True)[0][0] for text in X_test])
# Train a logistic regression classifier on the extracted features
lr_ml = LogisticRegression(random_state=42)
lr_ml.fit(X_train, y_train)
y_pred = lr_ml.predict(X_test)
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
Government News	0.55	0.14	0.22	79
Middle-east	0.30	0.17	0.21	54
News	0.78	0.94	0.85	553
US_News	0.35	0.15	0.21	53
left-news	0.40	0.18	0.24	262
politics	0.51	0.71	0.59	370
accuracy			0.62	1371
macro avg	0.48	0.38	0.39	1371
weighted avg	0.58	0.62	0.58	1371

```
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

From the above table you can see that roberta_base models gives somewhat decent accuracy. However, it is considerably less than our logistic regression model on all fronts.

b) Training an end to end classifier using the trainer function. The training parameters were:

learning rate = $1e-4$,
epochs = 1,
batch_size = 16, and
no weight decay

The results of the predictions on the validation set are as follows:

```
***** Running Prediction *****
  Num examples = 1371
  Batch size = 16
Accuracy: 0.4033552151714077
Precision: 0.16269542960597264
Recall: 0.4033552151714077
F1 Score: 0.23186635549874063
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and Recall are ill-defined and
_warn_prf(average, modifier, msg_start, len(result))
```

Here we can see that the accuracy dropped even further, suggesting that the chosen parameters were not good.

c) Now we will train 3 more end to end classifiers with different random parameters and test it on the validation dataset once again.

Classifier 1: Firstly, let us test what will happen to the accuracy if we reduce the learning rate and increase the batch size and number of epochs. This should equip our classifier better to predict the labels as it has more time to learn with a smaller learning rate which may lead to smoother training and a better convergence.

learning rate = $1e-5$,
epochs = 2,
batch_size = 32,
no weight decay

```

Epoch Training Loss Validation Loss
1      No log      1.436965
2      No log      1.175844

**** Running Evaluation ****
Num examples = 1371
Batch size = 32
**** Running Evaluation ****
Num examples = 1371
Batch size = 32

Training completed. Do not forget to share your model on huggingface.co/models =)

**** Running Prediction ****
Num examples = 1371
Batch size = 32
Accuracy: 0.5572574762946754
Precision: 0.38346937844570533
Recall: 0.5572574762946754
F1 Score: 0.45359000617128614
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
_warn_prf(average, modifier, msg_start, len(result))

```

As expected, the accuracy increase by quite a bit. This means we are moving in the right direction. Now let us modify the parameters a bit further to see if we can get even higher accuracy.

Classifier 2: Let us try and reduce the learning rate even more. However, lets not go overboard with it as may get stuck in a local optima if the learning rate is too small.

learning rate = 1e-6,
epochs = 3,
batch_size = 4
no weight decay

```

Epoch Training Loss Validation Loss
1      1.362600      1.250309
2      1.214200      1.142811
3      1.152400      1.120643

Accuracy: 0.5681983953318746
Precision: 0.478799649450555
Recall: 0.5681983953318746
F1 Score: 0.48453361554423185
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
_warn_prf(average, modifier, msg_start, len(result))

```

Another slight increase in accuracy, however, the precision still increased quite a bit. One can now assume that increasing the number of epochs and reducing learning rate by a little is helping our model.

Classifier 3: This time let us reduce the learning rate even further and mess with the batch size. We will test it again on the validation dataset

learning rate = 1e-7,
epochs = 1,
batch_size = 8
no weight decay

```
/usr/local/lib/python3.9/dist-packages/transformers/optimization.py:391: F
warnings.warn(
[515/515 03:52, Epoch 1/1]

Epoch Training Loss Validation Loss
1 1.140200 1.115578

Accuracy: 0.5689277899343544
Precision: 0.47509120017153394
Recall: 0.5689277899343544
F1 Score: 0.48609696265232033
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:
_warn_prf(average, modifier, msg_start, len(result))
```

It looks like our model is saturated, reducing the learning rate further will not help us. Due to memory and GPU issues, we are unable to test it further. However, it is worthwhile to note that tuning the parameter can still lead to an increased accuracy of our end to end classifier based on Roberta.

Also, a hit and trial method of finding the correct parameters is not ideal and very computationally expensive. More sophisticated methods should be used in order to find the correct parameters to get the most out of the model.

Here are the results of the three classifiers in a table along with their respective hyper parameters.

	Classifier 1	Classifier 2	Classifier 3
Accuracy	0.557	0.568	0.568
Precision	0.383	0.478	0.475
Recall	0.557	0.568	0.568
F1-score	0.453	0.484	0.486
Parameters	lr=0.00001, epochs=2, batch_size=32	lr=0.000001, epochs=3, batch_size=4	lr=0.0000001, epochs=1, batch_size=8

d) Using end to end models was definitely a better approach in terms of performance metrics like accuracy, precision, recall and f1 score as one can clearly see a jump in performance in the end to end models.

These approaches are different from each other in terms of the pipeline used. In part a, we used a feature extraction pipeline from HuggingFace. We only used the first context vector for each document which are then passed into a logistic regression model.

In the end to end classifiers, we have used the “trainer” function from the HuggingFace library. This approach is more complex as it requires optimizing the model using the parameters and fine tuning them. Because of our ability to fine tune the parameters, the end to end models are able to achieve higher accuracy than feature extraction models.

Q6.

a) The best approach from all the prior results would be the logistic regression (tfidf vectorization) with fine tuned parameters. Let us evaluate it on the test set and record the results.

These are the results on the validation dataset.

↳	precision	recall	f1-score	support
Government News	0.42	0.20	0.27	79
Middle-east	0.51	0.41	0.45	54
News	0.95	1.00	0.98	553
US_News	0.50	0.47	0.49	53
left-news	0.46	0.39	0.42	262
politics	0.56	0.67	0.61	370
accuracy			0.70	1371
macro avg	0.57	0.52	0.54	1371
weighted avg	0.69	0.70	0.69	1371
Accuracy: 0.703				
Macro-averaged Precision: 0.569				
Macro-averaged Recall: 0.523				
Macro-averaged F1: 0.537				

Here are the results on the training dataset.

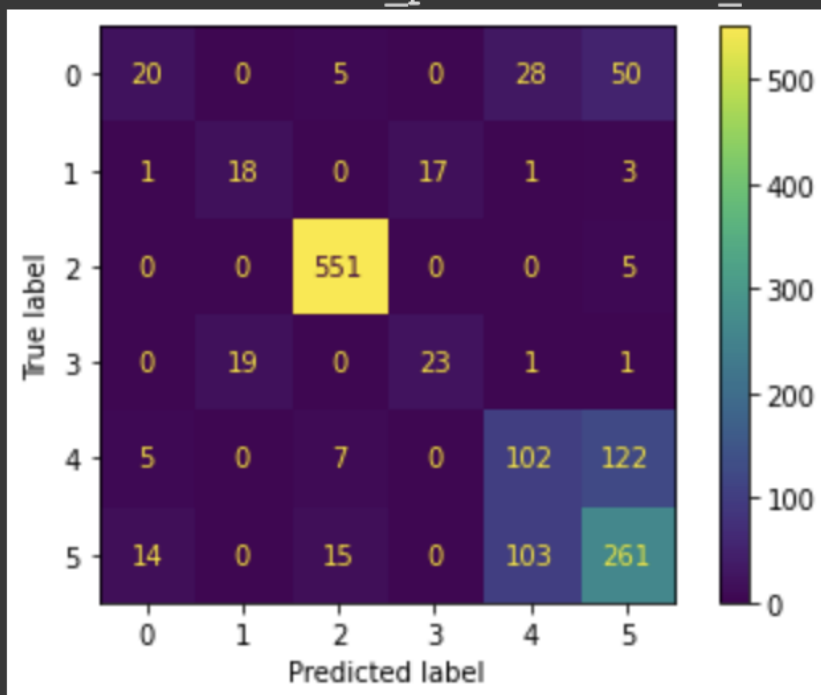
	precision	recall	f1-score	support
Government News	0.50	0.19	0.28	103
Middle-east	0.49	0.45	0.47	40
News	0.95	0.99	0.97	556
US_News	0.57	0.52	0.55	44
left-news	0.43	0.43	0.43	236
politics	0.59	0.66	0.63	393
accuracy			0.71	1372
macro avg	0.59	0.54	0.55	1372
weighted avg	0.70	0.71	0.70	1372

Accuracy: 0.711
 Macro-averaged Precision: 0.590
 Macro-averaged Recall: 0.542
 Macro-averaged F1: 0.554

As you can see, there isn't much difference in the results. This may be because of the fact that the validation dataset and training dataset are somewhat similar in terms of data distribution.

Below is a confusion matrix of the classifications on the test set.

```
[> <sklearn.metrics._plot.confusion_matrix.ConfusionMatrix
```



The labels are as follows:

- 'Government News': 0,
- 'Middle-east': 1,
- 'News': 2,
- 'US_News': 3,
- 'left-news': 4,
- 'politics': 5

b)

Let us analyse the confusion matrix step by step:

1) Let us look at the diagonal values of the confusion matrix. These values are high, which therefore suggests that the true positives are high. This gives us a little sense that the model is doing decent overall.

2) As we can see from the confusion matrix, label number 2 (News) has the highest amount of true positives. Whereas, label number one (Middle-east) has the lowest number of true positives.

3) Now if we look at the off diagonal values of the matrix, particularly at the square at 4x5 and 5x4, we can see that it is quite a large number. This means that the model is having difficulties in distinguishing between topics 4 and 5 which are “left-news” and “politics” respectively. Let's take a look at a few examples of a misclassification between these models:

- septemb 2016, stephen henderson, editor editor detroit free press, use power hi pen unjustli attack kid rock, liber fear rock might actual seriou run republican contend do-noth democrat senat debbi stabenow. gannett co. inc., newspaper parent company, say henderson behavior.
- make mistak it, nfl need show fan player seriou play football, owner go continu face catastroph financi losses. owner hire goodel social justic director dalla cowboy owner jerri jone led confer call 17 nfl owner thursday discu possibl halt commission roger goodel pend contract extension.
- left abl get away shame conserv easili past. outspoken conserv gop presidenti candid like donald trump, ben carson, ted cruz, mike huckabe carli fiorina start fight back liber medium outset presidenti elect season, open door conserv like steve cort fight back appropri boy, wa thi appropri time!msnbc joy reid, fill chri hayes, go head-to-head steve cortes, latino trump support donald trump, use term illeg describ illeg immigrants.

As you can see, even manually it is hard to determine whether a news article belongs to “left-news” or “politics”. The model is having difficulties with such classifications as the line that separates them is very thin.

The model is also having some problems in distinguishing between the labels 0 (“Government-News”) and 5 (“politics”). However, it is not as bad as the other pair.

c)

The final performance of the logistic regression (tfidf) model is:

	precision	recall	f1-score	support
Government News	0.50	0.19	0.28	103
Middle-east	0.49	0.45	0.47	40
News	0.95	0.99	0.97	556
US_News	0.57	0.52	0.55	44
left-news	0.43	0.43	0.43	236
politics	0.59	0.66	0.63	393
accuracy			0.71	1372
macro avg	0.59	0.54	0.55	1372
weighted avg	0.70	0.71	0.70	1372
Accuracy: 0.711				
Macro-averaged Precision: 0.590				
Macro-averaged Recall: 0.542				
Macro-averaged F1: 0.554				

The accuracy being 71% is not the best, however it turned out to be a decent model. This model will serve as a tool for classifying news by news channels. Therefore, the penalty of having a low accuracy is not very severe. Having some false positives or false negative will not have drastic effects on the industry. It might lead to some misinformation, but nothing that won't be easily rectified by the viewers themselves just by simply reading the news article.

According to the confusion matrix, the model is prone to misclassifying “left-news” as “politics” news and vice-versa. And in some cases, it may get confused between “Government-News” and “politics” as well. It also has a very high chance of correctly classifying articles as “News”, “left-news” and “politics”.

d)

Deployment of this model will have low to zero negative societal effects as it is merely an informational model. It can help a lot of news channels with their segregation process as long as it is also monitored with some human intervention. Misclassification of news does not have a severe backlash.

e) The main problem that we have faced while training this model is lack of data. The training data was unevenly distributed between the six labels. A more balanced data with a large number of rows for each label will be optimal to increase accuracy. A more thorough parameter tuning can also help in increasing the accuracy of the model. Trying out different model types can also be a viable approach to make this result better.

f) 45-50 hours.