Text-As-Data coursework

UID: 2823161S
Name: Sujan

Q1 - Dataset:

(a) This dataset for text sentiment analysis. The dataset has the form (1718, 4), and each row in it indicates a comment or review on a smart device, most likely an Amazon Echo Dot. Given that the dataset focuses on sentiment analysis, this would be a fantastic opportunity to learn how to properly analyze data in order to gain insights from it. The dataset was chosen for this reason. Moreover, the use of sentiment analysis is more widespread in:

- Product reviews
- Social media monitoring
- Customer service

Machine learning models can be trained using the automatic classification and labeling of this dataset to determine the sentiment of fresh comments and reviews with accuracy. This can be especially helpful in the mentioned applications since it enables businesses to quickly and effectively evaluate massive amounts of data and pinpoint areas for development.

b) Review Column : the actual text of the comment/review (text that will be used for classification)
Division Column : a categorical label indicating the sentiment of the text those are positive, neutral, or negative. (labels are to be predicted)

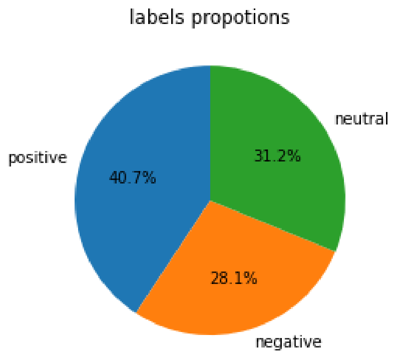
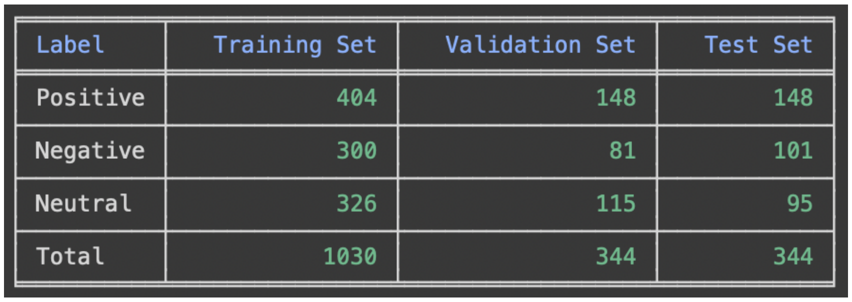


Fig. distribution of labels among dataset

Preprocessing : There was no need for preprocessing the label as it had exactly 3 labels. But Preprocessing operations are carried out on a review text using the function preprocess reviews. Emojis, numbers, and other non-alphanumeric characters were removed. Text was also converted to lowercase, tokenized, stop words eliminated, and non-nouns, verbs, adjectives, and adverbs filtered away. These phases are implemented by the function using a variety of libraries, including emoji, re, gensim, and wordnet. The function's overall goal is to clean and tokenize text data in preparation for additional modeling or analysis.

c) No,the dataset was not split into a training, validation and test set.



Label	Training Set	Validation Set	Test Set
Positive	404	148	148
Negative	300	81	101
Neutral	326	115	95
Total	1030	344	344

To ensure that the models trained on the training set can generalize successfully to the validation and test sets, it is crucial to have an even distribution of labels throughout the dataset splits.The model may be biased towards the majority class and may not perform well for the minority classes if the distribution of labels is unbalanced.

With comparable numbers of positive, negative, and neutral labels in each of the three sets, the label distribution throughout the training, validation, and test sets looks to be fairly balanced. The test set has an equal number of positive and negative examples but less neutral examples than the training set, which includes somewhat more positive examples than negative and neutral ones. The validation set has the fewest samples and distributes labels fairly evenly.

Q2 - Clustering:

a) Few examples of the documents assigned to each cluster, and the top 5 tokens with the highest magnitude in the corresponding centroid

Cluster 0: 93 documents	**Top 5 features with highest magnitude in centroid:** ['good', 'product', 'voice', 'recognition', 'buy']
	Examples: ● good product ● Good price
Cluster 1: 123 documents	**Top 5 features with highest magnitude in centroid:** ['work', 'properly', 'product', 'stop', 'wifi']
	Examples:

	● good product ensure will work excellently ● amaze product love product drawback work plug
Cluster 2: 41 documents	**Top 5 features with highest magnitude in centroid:** ['useful', 'feature', 'product', 'play', 'connect']
	Examples: ● gift mother birthday year shes lot fun discover many new feature great way ease parent iot ● useful general knowledge many things recognize kid voice nicely best gift son ● nice voice understand hindi english useful even control compatible devices voiceloving
Cluster 3: 1294 documents	**Top 5 features with highest magnitude in centroid:** ['product', 'like', 'nice', 'connect', 'buy']
	Examples: ● good product but bluetooth voice low ● could better rechargeable battery operate instrument ● good device learn
Cluster 4: 167 documents	**Top 5 features with highest magnitude in centroid:** ['quality', 'sound', 'good', 'voice', 'product']
	Examples: ● good sound respond time less best device information music ● good sound quality voice recognition follow need improvement

b) According to me, the clusters make sense. Each cluster stands out from the rest due to an unique set of top features.

For instance, Cluster 0 seems to have something to do with voice recognition and sound quality, but Cluster 1 seems to be more concerned with product connectivity and functionality. Cluster 4 appears to be focused on the sound and voice quality of the Echo Dot device, whereas Cluster 3 appears to be focused on overall product quality and satisfaction.Contrarily, Cluster 2 only comprises 63 papers and has more broad top features like "useful" "feature" and "play" more rely on the product feature. Overall, it appears that some clusters contain specific themes but not others, which shows that the clustering algorithm was successful in assembling related documents based on their content.

c) The confusion matrix displays the number of true labels that were properly predicted and the number that were wrongly guessed for each class. There are 5 classes in this instance, and according to the matrix, class 0 had 43 accurate predictions, class 1 had 416 correct predictions, class 2 had 51 accurate predictions, and classes 3 and 4 had no accurate predictions at all.

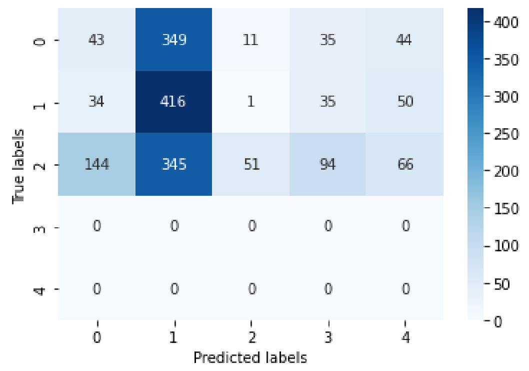


Fig. confusion matrix on target labels when k=5

d) The diagonal elements of the confusion matrix, which run from top-left to bottom-right, are what we can infer to be the number of properly predicted labels for each class. We observe that, respectively, clusters 0 and 1 appear to be more adept at detecting the "negative" and "positive" labels.

Among all labels, Cluster 2 seems to have the most misclassifications. Furthermore, clusters 0 and 1 exhibit a higher degree of label relatedness, with "negative" and "neutral" frequently misclassified within cluster 0 and "positive" and "neutral" frequently misclassified within cluster 1, respectively.The confusion matrix generally shows that there may be some association between the misclassifications of various labels within each cluster and that particular clusters may be better at detecting certain labels.

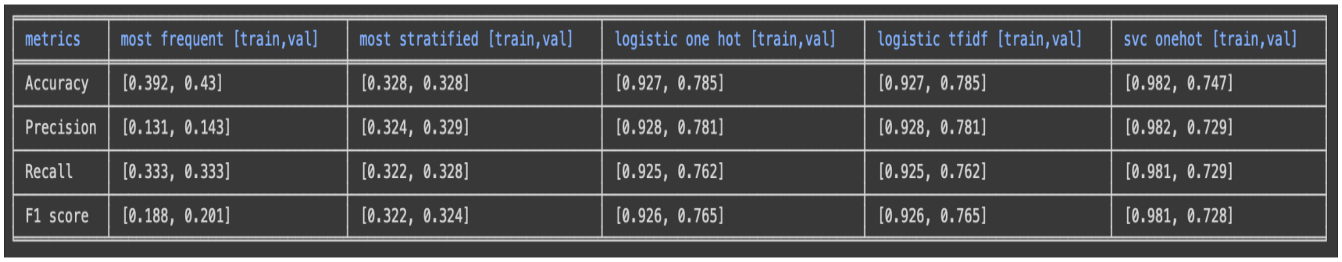
Q3 - Comparing Classifiers:

a)Analysis of the classifiers' effectiveness, their behaviors with respect to appropriate model "fit", dataset considerations, and classifier models.

- **Dummy Classifier with strategy="most_frequent":** The most prevalent class for each occurrence in the test dataset is predicted by this pretty simple classifier. With a low accuracy of 0.430, this classifier performs only marginally better than random guessing. The classifier creates a lot of false positive predictions since the precision score, which is only 0.143, is so low. The classifier misses a lot of instances of the positive class, as evidenced by the recall score's low value of 0.333. At 0.201, the F1 score is also incredibly low. This classifier is useless for any real-world application because it does not suit the training/test dataset.
- **Dummy Classifier with "stratified" as its strategy:** Based on the class distribution of the training data, this classifier generates predictions by randomizing them. Although still low, the accuracy score of 0.352 is marginally higher than that of the most popular classifier. The classifier makes less false positive predictions when compared to the most frequent classifier, as evidenced by the greater precision score of 0.343. Moreover, the classifier misses less occurrences of the positive class, as evidenced by the recall score's increase to 0.342. Although the F1 score of 0.341 is greater than that of the most popular classifier, it is still quite low. This classifier is useless for any real-world application because it does not suit the training/test dataset.
- **LogisticRegression with One-hot vectorization:** This linear classifier forecasts the class labels using logistic regression. The textual data is converted into a numerical representation that the classifier may use via one-hot vectorization. The accuracy score is good at 0.802, showing that the classifier is fit well to the training/test dataset. The classifier makes less false positive predictions, according to the accuracy score of 0.803, which is also quite high. The classifier misses some occurrences of the positive class, as evidenced by the recall score, which is lower at 0.783. The overall F1 score of 0.787 is an excellent one. This classifier performs well in real-world applications, and the one-hot vectorization method appears to be efficient.
- **LogisticRegression with TF-IDF vectorization (default settings):** This is similar to the previous classifier, however this time, TF-IDF vectorization is employed instead of one-hot vectorization. At 0.802 and 0.803, respectively, the accuracy score and precision score are identical to those of the prior classifier. Also identical, at 0.783, is the recall score. The F1 score is the same as the previous classifier at 0.787. This indicates that for this dataset, both vectorization methods perform equally well. For

real-world applications, this classifier works well, however the TF-IDF vectorization method offers no advantages over one-hot vectorization.

- **SVC Classifier with One-hot vectorization (SVM with RBF kernel, default settings):** This nonlinear classifier forecasts the class labels using the SVM method and an RBF kernel. The textual data is converted into a numerical representation that the classifier may use via one-hot vectorization. The accuracy score is 0.770, which is less than the classifiers created using logistic regression. The classifier makes more false positive predictions, as evidenced by the lower precision score of 0.756. The recall score is 0.752, which is marginally inferior to the classifiers produced by logistic regression. The F1 score is 0.753, which is similarly less than the classifiers from the logistic regression.



metrics	most frequent [train,val]	most stratified [train,val]	logistic one hot [train,val]	logistic tfidf [train,val]	svc onehot [train,val]
Accuracy	[0.392, 0.43]	[0.328, 0.328]	[0.927, 0.785]	[0.927, 0.785]	[0.982, 0.747]
Precision	[0.131, 0.143]	[0.324, 0.329]	[0.928, 0.781]	[0.928, 0.781]	[0.982, 0.729]
Recall	[0.333, 0.333]	[0.322, 0.328]	[0.925, 0.762]	[0.925, 0.762]	[0.981, 0.729]
F1 score	[0.188, 0.201]	[0.322, 0.324]	[0.926, 0.765]	[0.926, 0.765]	[0.981, 0.728]

Fig. evaluation metrics on training and validation dataset respectively

Comparison between all the classifiers:

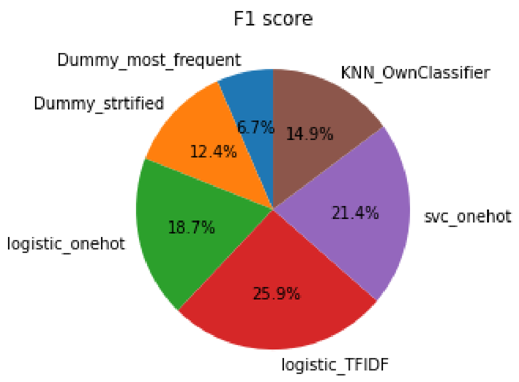


Fig. Contribution of each classifier

Comparing the two baseline models—the Dummy Classifier with strategies of "most frequent" and "stratified"—to the logistic regression models and the SVC Classifier with one-hot vectorization, the baseline models don't fare well. The logistic regression models with one-hot and TF-IDF vectorization techniques offer good accuracy,precision, and F1 scores, making them effective for practical use cases.

The LogisticRegression models with either one-hot or TF-IDF vectorization are the best performing models out of all these. Both of these models have high accuracy, precision, and F1 scores, indicating that they fit well to the training/test dataset and are effective for practical use cases.

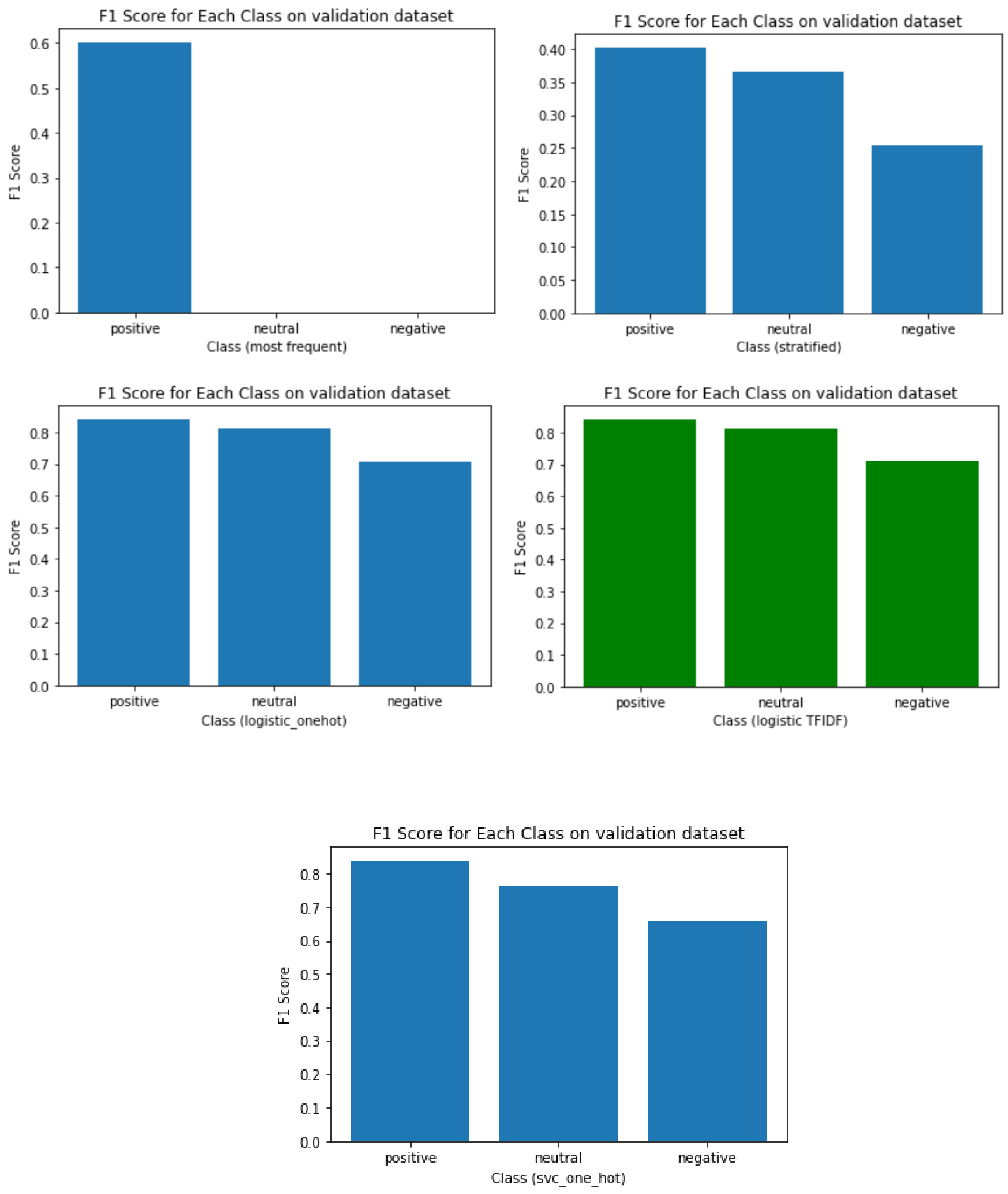
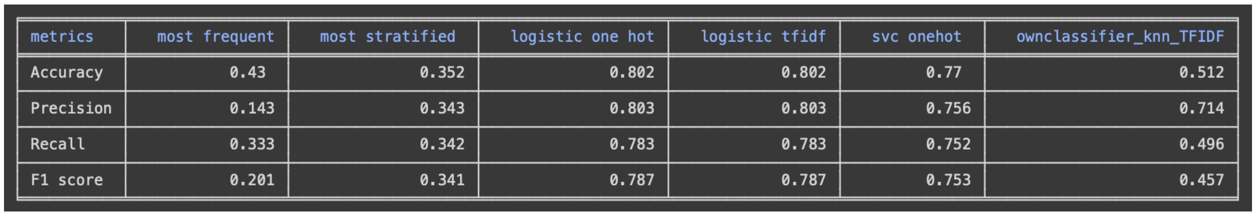


Fig.Barchart graph with the F1 score for each class for all classifiers + best classifier (green color)

b) The non-parametric machine learning algorithm K-Nearest Neighbors (KNN) is used for classification and regression tasks. The popular vectorization method known as

TF-IDF (Term Frequency-Inverse Document Frequency) turns text into numerical features. Words are given weight based on how frequently they appear in a document and how many documents in the corpus use that word. This aids in identifying a word's significance in a document and separating it from less important ones.

Overall, the KNN and TF-IDF vectorization combination is suitable for text classification applications with high-dimensional feature spaces and modest dataset sizes. In addition to being utilized in ensemble approaches to enhance overall model performance, it can serve as a useful baseline model to compare with more complicated models. Ultimately after trying all other classifiers such as Randomforest, Decision trees,Naive bayes etc, KNN with TF-Idf vectorizer performed best.



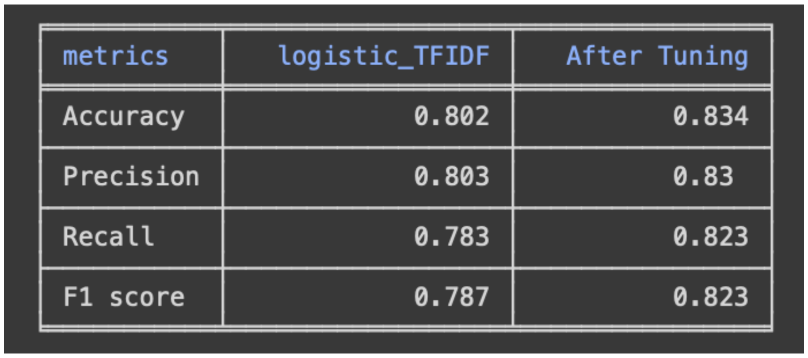
metrics	most frequent	most stratified	logistic one hot	logistic tfidf	svc onehot	ownclassifier_knn_TFIDF
Accuracy	0.43	0.352	0.802	0.802	0.77	0.512
Precision	0.143	0.343	0.803	0.803	0.756	0.714
Recall	0.333	0.342	0.783	0.783	0.752	0.496
F1 score	0.201	0.341	0.787	0.787	0.753	0.457

Fig- Table comparing the metrics of different classifier with my own implemented classifier

Q4 - Parameter Tuning:

1. Regularisation C value : I tried with the range [1,2,3,4,5,6] . Lower the c value higher is the regularization. Hence **“2” was performing best on the model.**
2. Vectorizer - Parameters:
   a. sublinear_tf = True or false
      i. The TfidfVectorizer function uses the raw term frequency values rather than applying a sublinear scaling function to them when the sublinear TF parameter is set to False. This indicates that the TF-IDF algorithm directly utilizes each phrase frequency value. Hence model performs best when **sublinear_tf -False**
   b. max_features:
      i. The vectorizer will only maintain the top 1705 most often occurring words in the corpus when the max features argument in the **TfidfVectorizer function is set to 1705**. The other words will be discarded. These top terms are chosen according to their document frequency, or the quantity of papers in which a specific word appears.
3. Another parameter of your choice from the classifier :
   a. min_df : **(set to 4)** When the TfidfVectorizer function's min df parameter is set to 4, the vectorizer will reject words that appear in less than four of the corpus's documents. In other words, when constructing the TF-IDF matrix, only terms that exist in at least 4 papers will be taken into account. This can be helpful for removing uncommon terms from the corpus that might cause noise.

All these parameters mentioned above helped me in improving the metrics scores. But there are some more parameters such as Penalty in logisticRegression, max_df in Vectorizer etc but all these were degrading the score.

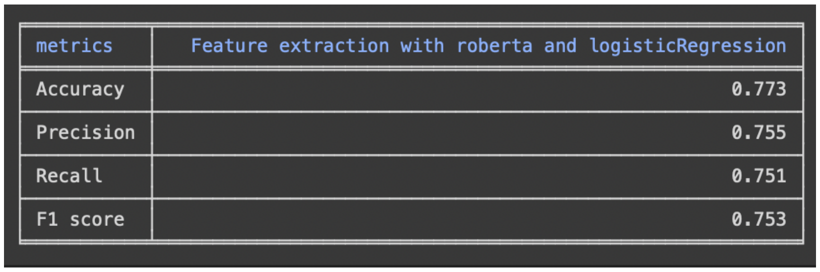


metrics	logistic_TFIDF	After Tuning
Accuracy	0.802	0.834
Precision	0.803	0.83
Recall	0.783	0.823
F1 score	0.787	0.823

Fig - Table comparing the metrics of logisticRegression before and after tuning

Q5 - Context vectors using BERT :

(a) The algorithm uses RoBERTa-based feature extraction to train a logistic regression model for sentiment analysis. It extracts features from the input text using the pre-trained RoBERTa model and tokenizer and then trains a logistic regression model on the features to predict the sentiment labels of the input text. Accuracy, precision, recall, and F1 score are among the metrics that are reported by the algorithm while evaluating the model's performance on a validation set. The outcomes imply that the model performs rather well on the validation set.

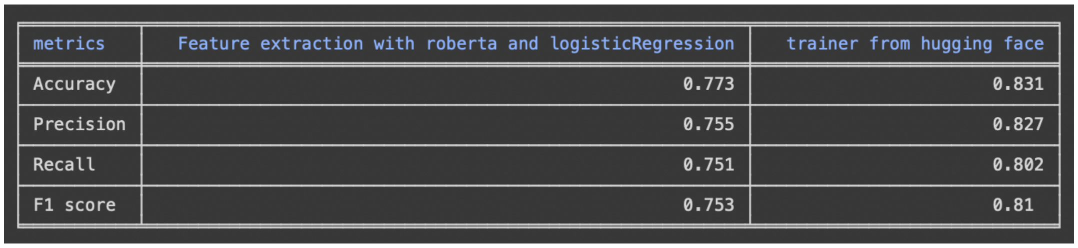


metrics	Feature extraction with roberta and logisticRegression
Accuracy	0.773
Precision	0.755
Recall	0.751
F1 score	0.753

Fig. Hugging face transformers with logistic regression model

(b) A high-level interface for training and assessing deep learning models is provided by the Trainer class from the Hugging Face library. This strategy, which involves feature extraction for the initial context vector for each document, is frequently applied to classification tasks at the sentence and document levels. Each sentence or document should be represented as a fixed-length vector so that it may be fed into a classifier.

Comparison: In terms of accuracy, precision, recall, and F1 score, the Trainer model from Hugging Face exceeds the logistic regression using RoBERTa base model. All four metrics had superior results for the Trainer model. In particular, the accuracy of the Trainer model is 0.831, which is higher than the accuracy of the logistic regression with RoBERTa base model, which is 0.773. Moreover, the Trainer model outperforms the logistic regression with RoBERTa base model in terms of precision, recall, and F1 score.

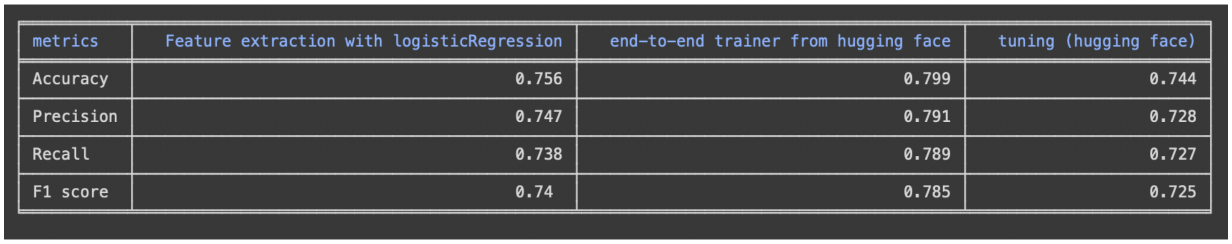


metrics	Feature extraction with roberta and logisticRegression	trainer from hugging face
Accuracy	0.773	0.831
Precision	0.755	0.827
Recall	0.751	0.802
F1 score	0.753	0.81

Fig. Comparison between roberta model with logisticRegression vs Trainer from hugging face.

(c) A model's performance can be significantly affected by altering the hyperparameters, including learning rate, batch size, and number of epochs. In yourmy situation, performance has declined as a result of changing the hyperparameters from a learning rate of 1e-4, batch size of 16, and 1 epoch to a learning rate of 5e-5, batch size of 32, and 1 epoch.This could've happened because of following reasons:

- Learning rate: Reducing the learning rate could cause the model to converge more slowly, taking longer to arrive at the ideal weights.
- Batch_size: If the batch size is too big, it could lead to convergence to a less-than-ideal solution or even divergence.
- Epochs: A model may not have had enough time to converge to the ideal solution if the number of epochs is too low.



metrics	Feature extraction with logisticRegression	end-to-end trainer from hugging face	tuning (hugging face)
Accuracy	0.756	0.799	0.744
Precision	0.747	0.791	0.728
Recall	0.738	0.789	0.727
F1 score	0.74	0.785	0.725

(**d) Best performed model is : HuggingFace library with the ‘roberta_base’ model**

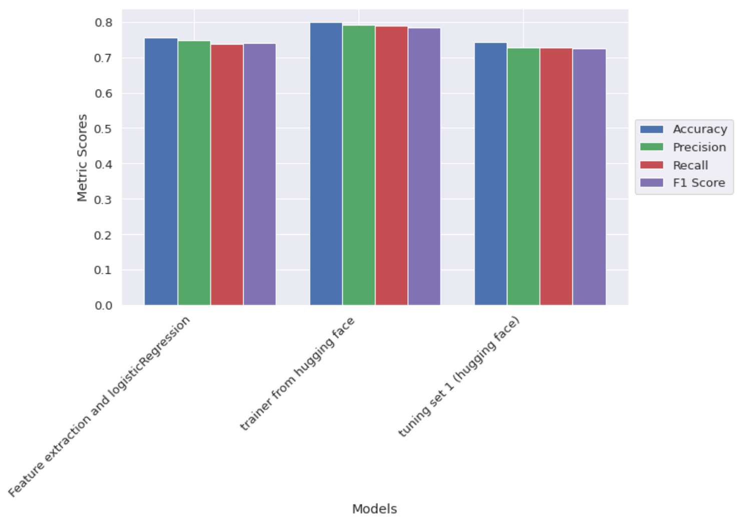


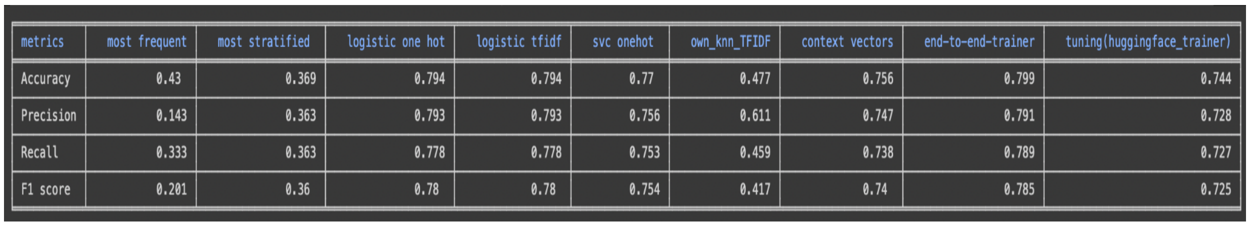
Fig. Evaluation metrics between context vector vs end-to end trainer vs tuned trainer

The first approach makes use of a fixed feature extraction technique before moving on to a different logistic regression classifier, which is the main distinction between the two methods. The second method, in contrast, trains the model from beginning to end, which means that the model picks out pertinent features and classes the documents at the same time.

The second strategy trains the model from beginning to end, giving it more freedom to learn and adapt to the particular job, which may be the main cause of any performance differences between the two approaches. In order to improve performance , the second method also enables more precise adjustment of the model's hyper - parameters (such as learning rate, epochs, batch size, etc.). In contrast, the first method uses a fixed feature extraction method and a separate classifier, which might not be as optimized for the particular task.

Q6 - Conclusions and Future Work :

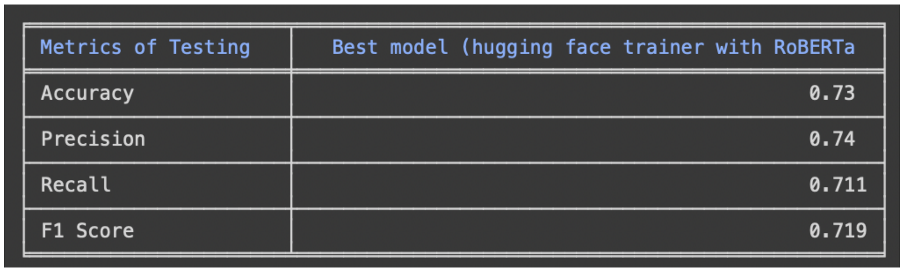
Now let's find out which is the Best model out of Q3/Q4/Q5 by means of a table which contains evaluation metrics of all models/ classifiers



metrics	most frequent	most stratified	logistic one hot	logistic tfidf	svc onehot	own_knn_TFIDF	context vectors	end-to-end-trainer	tuning(huggingface_trainer)
Accuracy	0.43	0.369	0.794	0.794	0.77	0.477	0.756	0.799	0.744
Precision	0.143	0.363	0.793	0.793	0.756	0.611	0.747	0.791	0.728
Recall	0.333	0.363	0.778	0.778	0.753	0.459	0.738	0.789	0.727
F1 score	0.201	0.36	0.78	0.78	0.754	0.417	0.74	0.785	0.725

Fig. Evaluation metrics of all classifiers

(a) Again from the table given above we can infer that the best model form Q3/Q4/Q5 is the **HuggingFace library with the 'roberta_base' model.** The evaluation metrics of the test data set is given below:



Metrics of Testing	Best model (hugging face trainer with RoBERTa
Accuracy	0.73
Precision	0.74
Recall	0.711
F1 Score	0.719

Fig. Evaluation metrics for test dataset

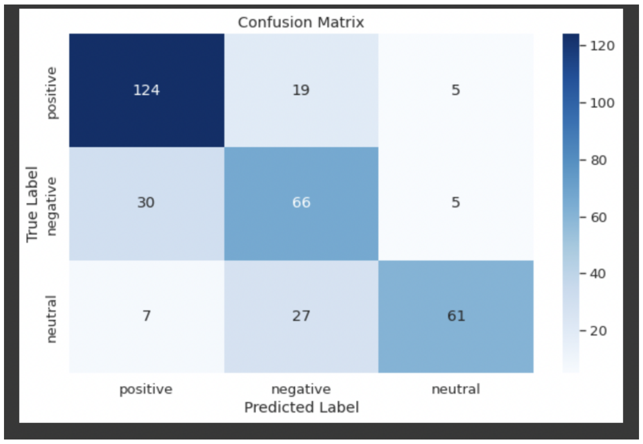


Fig.confusion matrix of the classifications on the test set.

(**B)** By examining the confusion matrix, we can see that the model generally works admirably, properly classifying the majority of the data. The following patterns and tendencies can be seen, though:

- Classifying neutral samples accurately is where the model struggles the most. To be more specific, 7 samples that were truly positive were classed as neutral, while 27 samples that were actually negative.
- The model frequently confuses items that are negative and neutral. Particularly, 5 negative samples were designated as neutral, while 19 negative samples were categorized as positive. Five neutral samples were also deemed negative.
- With only 5 positive samples misclassified as negative, the model does reasonably well at categorizing positive samples.

Error analysis: In order to undertake a more extensive error analysis, we may examine individual instances of incorrectly categorized samples in an effort to spot recurring patterns or themes in the text that might be confusing the model. For instance, we may examine the text of the negative samples that were incorrectly categorized as positive to determine whether there are any typical positive idioms or phrases that are generating misunderstanding. Similar to this, we may examine the text of the neutral samples that were incorrectly categorized as negative in order to identify any recurring negative phrases or expressions that the model may be detecting.

For example: I conducted an error analysis in a notebook and retrieved some examples from it.

```
Index:  3
Text:  thankful amazon product buy daughter tell amazon
True label:  positive
Predicted label:  negative
------------

Index:  10
Text:  like product lot time connect
True label:  neutral
Predicted label:  negative
------------
```

More examples is in notebook

(c) The model attained an accuracy of 0.799, precision of 0.791, recall of 0.789, and an F1-score of 0.785 based on the metrics offered. These measures appear to show a passable but not outstanding performance from the model. It is significant to remember that the model's performance may change depending on the particular use case.

The adoption of this machine learning pipeline can be significantly impacted by false positives and false negatives. False positives could cause positive sentiment to be predicted incorrectly, which could lead to decisions being made based on the sentiment analysis results that aren't accurate. False positives could result in inaccurate judgements being made about the product's popularity, for instance, if this model were being used to predict the mood of customer reviews for a product. False negatives, on the other hand, can cause people to overlook bad emotion, which might also result in poor decisions. False negatives, for instance, may cause a business to miss significant negative sentiment if the model were being used to monitor social media sentiment for the brand.

According to the confusion matrix, the model exhibited a lot of false negatives for the 'negative' class, with 27 occurrences of negative sentiment being misclassified as 'neutral. This might be a sign that the model has trouble telling the difference between neutral and negative sentiment, possibly as a result of the similarity in language between these two categories. Also, the model produced 30 instances of neutral emotion that were mistakenly labeled as "positive," which is a surprisingly high amount of false positives for the "neutral" class. This could be a sign that the model predicts sentiment too optimistically, which could result in decisions being made based on the sentiment analysis results that aren't accurate.

(d) Indeed, the implementation of this system might have detrimental impacts on society. If the training data used to create the model are biased, one possible negative effect is that the system can reinforce biases and discrimination. As a result, the system might unfairly penalize or discriminate against some groups of people.

A lack of human oversight and responsibility may result from the usage of automated systems like this, which could have detrimental effects when decisions made by the system have a large impact on people's lives.

(e) Here are some possible actions that could be made to raise the system's capacity for classification:

- More training data can help the model generalize and perform well on new data. Gathering and adding more training data can enhance the model.
- Testing out various previously-trained models: To see if other pre-trained models are more appropriate for the task at hand, they can be used and compared to the current model.

(f) I took almost 30 to 40 hours and tried to understand the topic, tried to apply some uniqueness in the project and to improve the structure of the report .

Q7 - Research Paper Report :

The topic that i chose is “**Enriching Word Vectors with Subword Information**”

Background and context of the paper :

The author explores various strategies to make neural network models faster, and it highlights the Transformer model as an example. Unlike other models that rely on sequence-aligned RNNs or convolution, the Transformer utilizes self-attention to compute input and output representations. Self-attention is a mechanism that calculates the relationship between different parts of a sequence, allowing the Transformer to more efficiently learn dependencies and reduce the computational burden. The paragraph emphasizes the superiority of self-attention over other models such as convolutional neural networks and end-to-end memory networks.

Contributions of this paper :

The Transformer model, a competitive neural sequence transduction model. In this model, an encoder-decoder structure is used, in which the encoder converts an input sequence into a series of continuous representations, and the decoder produces an output sequence of symbols based on the encoded representations. The encoder and decoder of the Transformer use layered self-attention and point-wise, fully connected layers. A stack of N = 6 identical layers makes up the encoder, and a stack of N = 6 identical layers makes up the decoder with the addition of a third sub-layer that handles multi-head attention over the encoder stack's output.

The attention function of the Transformer converts key-value pairs and queries into an output that is calculated as a weighted sum of values based on a compatibility function. The "Scaled Dot-Product Attention" function uses matrices of queries, keys, and values to carry out its operations.

The Transformer performs the attention function simultaneously while continuously projecting queries, keys, and values to several learned subspaces. This enhances sequence modeling by allowing the model to pay attention to various sorts of input coming from various points.

Critique :

The authors' conclusions are supported by their data. The big transformer model they developed outperforms previously reported models on the WMT 2014 English-to-German and English-to-French translation tasks, achieving new state-of-the-art BLEU scores. Even their base models perform better than previously published models and ensembles, while being much cheaper to train. The authors provide detailed information on the hyperparameters and training process used, which adds to the credibility of their results.

A positive aspect of the authors' approach is their use of attention mechanisms instead of the commonly used recurrent layers in encoder-decoder architectures. This results in faster training times and improved translation performance for the Transformer model. The authors also provide detailed information on the training process and hyperparameters, which enhances the credibility of their results.

A potential downside of the authors' approach is their limited evaluation of the Transformer model, which only involved text-based translation tasks. This raises doubts about its performance on other tasks that involve different modalities. Although the authors have expressed interest in extending the model to other tasks, it is uncertain how well it will perform. Another drawback is the absence of a detailed analysis of the limitations of their approach or areas for improvement, which could benefit future research