University of Glasgow

School of Computing Science

COMPSCI5106 - TEXT AS DATA

AAYUSH NAGPAL

School of Computing Science

Sir Alwyn Williams Building

University of Glasgow

G12 8RZ

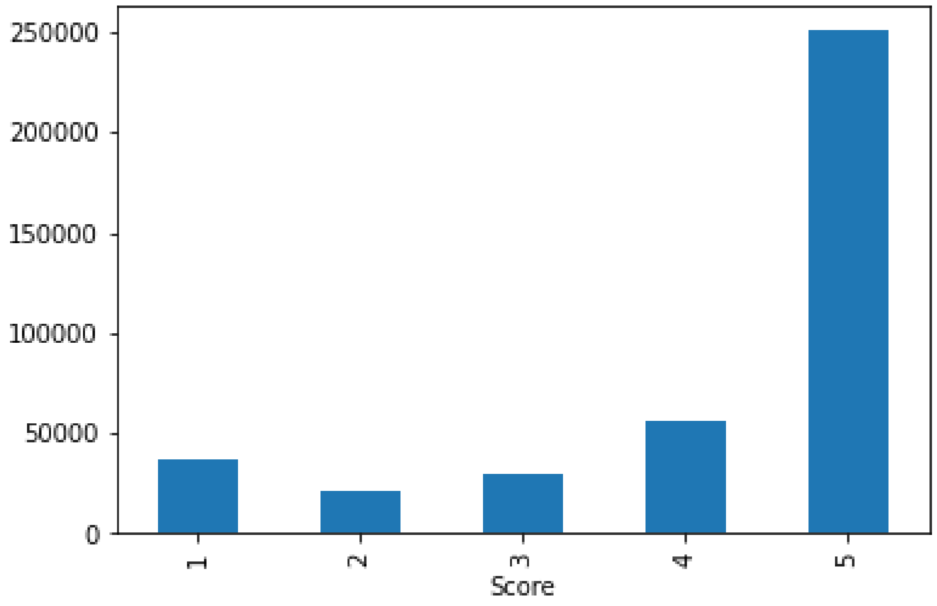
Q1 - Dataset

(a) What is your dataset, and why did you select it? How might automatic classification/labelling of your dataset be used in practice?

As part of this exercise, the aim was to demonstrate the ability of text processing techniques, and applications to text classification. For this, I have selected the Amazon Fine Foods review dataset. This dataset contains more than 350,000 customer reviews on products sold on the Amazon website. The motivation behind selecting this is to understand how we can extract meaningful information by analyzing customer reviews. In our case, the most useful portion of the data is the review and the score given by the customer for products. Automatic classification of such a dataset can help a company like Amazon in understanding the demands and requirements of their customers as well the current trends. Automatic classification can help the company in detecting products that have very poor reviews and then act on improving the customer experience. This can also help in building a product recommendation system.

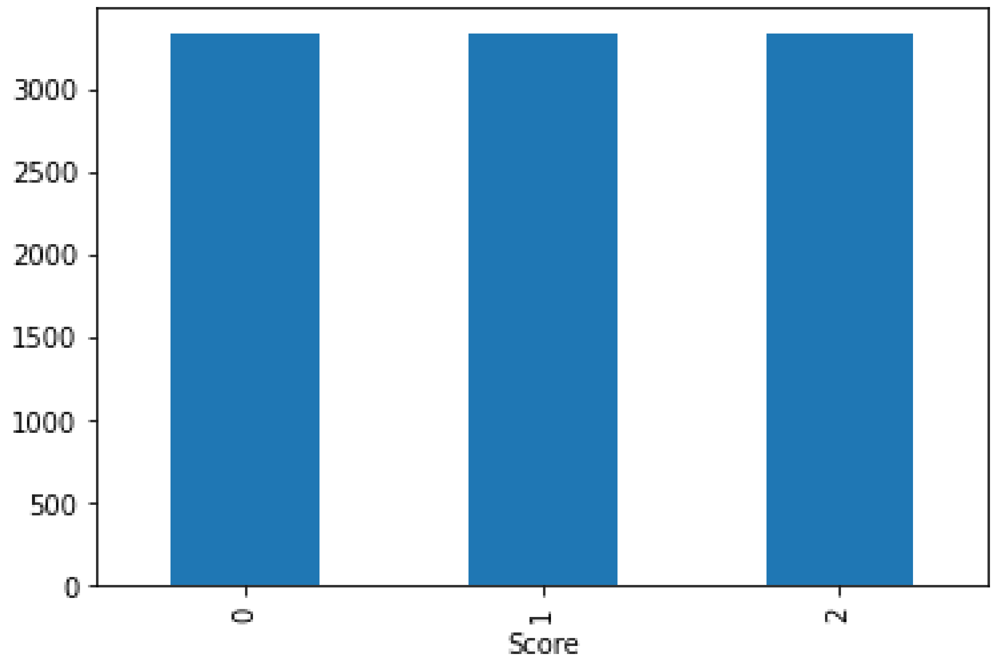
(b) Provide a summary of the labels and input text to be used. What labels are to be predicted? Did you need to do any pre-processing to create the labels or to make sure the number of labels was > 3 and <= 10? What is the text that will be used for classification?

The dataset contains a 'Score' column for each product. There are 5 unique values for this field as represented by the graph below [Graph 1]. This score highlights user satisfaction. If the score value for a product is 1 or 2 that means the customer is not satisfied with the food item. A score of 3 is for neutral emotion. If it's 4 or 5 then the customer is happy with the product. From Graph 1, we can observe that the dataset is highly skewed towards positive reviews and any model trained on such a dataset will not yield good results.



Graph 1- Distribution of Original Data

To solve the problem highlighted above, we will under-sample the dataset. As part of the coursework specification, the dataset cannot be of more than 10000 records. Thus, we will under-sample the dataset such that the data is distributed equally across all 3 labels ([0,1,2]).



Graph 2 – Data distribution after sampling and relabeling.

The 'Score' values are directly available in the dataset. Still, for this assessment, we have relabeled the product scores such that score with values 1 or 2 are marked as 0, 3(neutral emotion) is relabeled as 1, and products with scores 4 or 5 are marked as 2 [Graph 2].

Also, as part of data pre-processing, I am removing the following parts from product reviews:

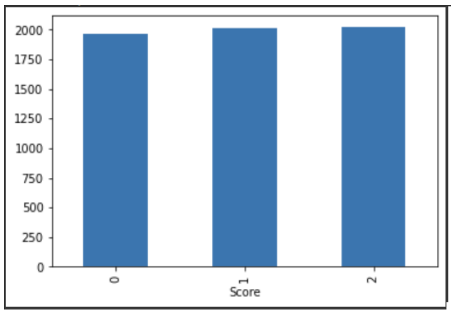
- extra spaces between text
- punctuations
- emojis, symbols, or flags
- email addresses if added by the customer
- newline characters, and single quotes
- convert text into lower case

(c) Is the dataset already split into a training, validation and test set? If not, use a 60/20/20% split to create the training, validation and test set. Provide a table with the label counts for each split of the dataset and comment on the distribution across labels and across data splits. Be sure you use the same splits throughout the entire report.

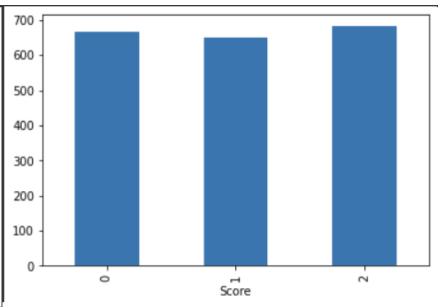
The given dataset contains 9999 records. Since the data is not divided into 3 parts. We will Use train_test_split from scikit-learn divide the data into training, validation and testing set in the ratio of 60:20:20 [Table-1] and Graph A, Graph B & Graph C.

	Dataset	#
0	Training	5999
1	Validation	2000
2	Testing	2000

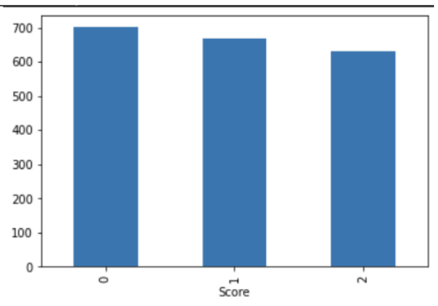
Table 1 – Dataset split.



Graph A – Train Dataset



Graph B – Valid Dataset



Graph C – Test Dataset

Q2 – Clustering

(a) When using k=5 clusters, give a few examples of the documents assigned to each cluster, and the top 5 tokens with the highest magnitude in the corresponding centroid.

For **Task 2** I wrote an implementation of Kmeans by defining a class for KMeans. The value of K for this task was set to 5 (as specified). Since, the implementation uses Euclidean distance to form clusters, it was observed that each cluster contained customer reviews related to similar topics. Below is an example of few selected documents.

```
Cluster 0:
      1. crushed good product good price chips survived intact crumbs pieces
tasty still way many crumbs pieces smaller bags travelled better less damage
      2. bland great loved mcdougalls soups tried hard time recommending one
pretty bland size cup bit smaller others worth price like texture soup
however
      3. couldnt get cat care cat scratched maybe twice never maybe placed
weight base like mentioned stop sliding around would used ended making diy
post sisal rope wood tube cardboard carpet gets sold cats may like built
exactly like would expected cardboard
      4. excellent product quite simply wonderful flax seed oil one often
drawn product reviews known reputation spectrum products filtered oil use
```

daily needs important deciding brand due auto immune condition like proportion omega 3s product flax seed oil best used within two months opening always keep refrigerated purchased smaller containers past whole foods seemed like frugal option happy product sent protection heat arrived promptly storage requirements

5. reliable reliable brand shopping major warehouse store gee could lol discovered sold loaded cart tones buy gourmet brands like shelf full tones products everyday use backup flavorful long lasting considering dried item close fresh get

Cluster 1:

1. best deal could find searched internet local stores best deal k cups amazon beat average 0 57 per cup shipping eonomical far k cups go also really great tasting coffee strong weak perfect always great shipping always time usually arrives early
2. tastes like burned melted plastic coffee never met coffee creamer couldnt stand tried stuff horrible taste reminds burned melting plastic smells took 2 sips absolutely disgusted threw coffee brewed fresh cup skipping awful creamer never buy stuff
3. good k cup instant gratification reviewers seem expect italian coffee house products every product made k cups nice cappuccino french vanilla flavored drink well find gas stations good others get gas station make bad go italy find makes acceptable cup using middle lowest setting remember less stronger 60 cal 40 fat 4 5g fat sat 9 carb 5g sugar 0g fiber
4. pumpkin spice k kups great flavor aroma always ease use kuerig fresh cup coffee less 2 minutes wish available year round vs fall
5. really good stuff good hot brewed coffee disappointing easy stomach us drink much coffee ware change coffee urge ever

Cluster 2:

1. way sweet trying types products use keurig coffee maker thought caramel cappuccino would good choice sweet couldnt drink throw sweet tooth much seems sweet wasted money throwing well grove square hazelnut cappuccino bought waste money unless like super sweet
2. alright great first taste think great leaves slight chemical taste mouth would buy product straight teas tasted much better latte style
3. simply awful water tastes like sewage dont hyped buying like going back research nothing product bodies need except water find kitchen looks horrible tastes worse feel need drink black brown water put water glass go garden dig dirt add
4. extremely sweet used sugar free diet drinks think find flavor overly sweet husband loves stomach however new sugar free drinks probably great place start
5. okay dont get big deal first nearly delicious regular chips dont even satisfy chip craving taste okay slightly carboardy styrofoamy e qualities want chip

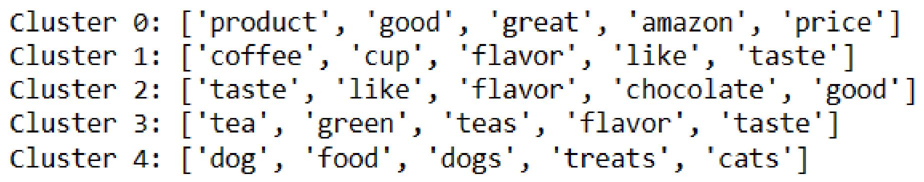
Cluster 3:

1. slaking thirst titivating taste one better tasting green teas tried somewhat bitter tannic tongue ways like weak black tea green tea orange tan color lacking grassy taste green teas carry said good value arrived promptly fine everyday drink wouldnt serve company seems work best water 160 170 degrees steeped 1 minute yield multiple 4 cup carafes course day dont let sit carafe thermos 3 hours gets bitter green teas found
2. try gloria jeans peppermint herbal tea instead larger world favorite mint tea bigelows plantation mint unfortunately thats available keurig world trying replicate plantation mint bought four sample packs mint options directly keurig testing gloria jeans version best mints far problem timothys strong twang front noticeable unpleasant chemical aftertaste tastes bit like

```
chewing mint leaf whole even cut lot water iced tea still sharp quality
hesitate add lest make sound crazy also feels like forming film tongue throat
swallow ive tried dozen keurig flavors iced tea none others make film try
must take someone whos actually compared better options mint tea keurig world
make iced tea easily keurig need strong glass wont break taken boiling
freezing crate barrels excellent large working glass strongest ive found
still looks good 21 ounce glass put sweetener 1 75 teaspoons splenda 1 25
husband brew medium cup tea add ice almost fill glass ice melt easily end 1 4
total volume stir drink quality ice matters since melted part become least
half drink youd like try flavored iced tea mint favorite gjs passion fruit
stock cant link available try close second place tea gloria jeans coffees tea
k cup mango flavored black tea keurig brewers 24 count boxes pack 2
      3. favorite tea ive always loved original version tea husband typically
look caffeine free teas although strong flavorful original still spicy
flavorful available live happy amazon offers price better stores
      4. great tea pictured bought tea thinking normal acai mango flavor
especially since picture shows fact read title carefully ice version flavor 6
boxes wrong flavor id rate 4 5 stars since tasty tea prefer original flavor
minus two stars false advertising
      5. great tasting tea good news tea tastes great overpowering bleach
taste like lipton box bought bag thing box got wasnt exact one picture less
colorful doesnt really matter also provides 0 71 ounces less tea total 2 84
less ounces boxes include 5 29 ounces tea instead 6 ounces box pictured
usually expect get exact item one pictured give 3 stars reason otherwise
great tea

Cluster 4:
      1. great dogs teeth started dogs dentastix really made difference teeth
plus loved local walmart stopped carrying found amazon price walmart plus tax
shipping buy bulk problem since two dogs get one daily figured would six
eight weeks worth pleased dentastix usual amazon com came
      2. reaction picky dog sale petco gave try theyre big give frequent
clicker treats 35 lb dog cutting half right even though theyre heavily
scented theyre favorite shed rather 3 pounds lamb lung baa baa qs dogswell
vitality dog treats chicken 6 ounce pouches pack 6 day
      3. good stuff big dog 100 lbs eats less 3 cups day always satisfied
bought struggling diarrhea poops always firm
      4. bully springs excellent bully springs one big small yorkie chew gave
neighbors larger dog able chew skinnier one seemed enjoy wish came skinnier
easy chew smaller dogs never know place order im going get im going keep
looking highly recommend larger dogs
      5. great food supervised lt 1 yr worked great us kid learning eat always
supervising food kinda dangerous said babies good choking stuff back doctor
mentioned confuse cant breathe choking much food choking 2nd 99 9 fine theyll
choke back 1st worry wind pipe sized things like whole peanuts marbles grapes
tomatoes etc hes fussy give baby one makes happy long enough change cheer
ways
```

(b) Based on (a), do the clusters make sense? Are there certain topics that appear in some but not others?

To see the top 5 documents from each cluster refer the Jupyter notebook. As we can observe from output above each cluster has documents pertaining to specific topics.

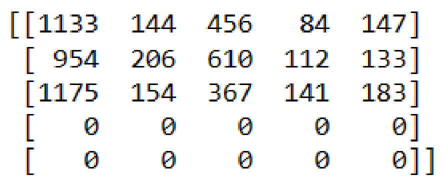


```
Cluster 0: ['product', 'good', 'great', 'amazon', 'price']
Cluster 1: ['coffee', 'cup', 'flavor', 'like', 'taste']
Cluster 2: ['taste', 'like', 'flavor', 'chocolate', 'good']
Cluster 3: ['tea', 'green', 'teas', 'flavor', 'taste']
Cluster 4: ['dog', 'food', 'dogs', 'treats', 'cats']
```

Image 1 – Cluster Tokens

Observing the tokens and top 5 documents, it is obvious that cluster 4 is about animal food and treats specifically for cats and dogs. It also includes sentences with baby food as well. Whereas cluster 3 is all about types of tea and tea flavours reviewed by customers. It slightly difficult to find a correlation between top 5 token and sentences of cluster 2. Cluster 1 contains reviews related to coffee and different types of coffee flavours and justifies tokens. Cluster 0 is little more generic than other a refers to general products reviewed by the customers.

(c) Construct a confusion matrix between the k=5 clusters and your target labels.



```
[[1133  144  456   84  147]
 [ 954  206  610  112  133]
 [1175  154  367  141  183]
 [   0    0    0    0    0]
 [   0    0    0    0    0]]
```

Image 2 – Kmeans (K=5) Confusion Matrix

The performance of a clustering method on training dataset with k=5 clusters is seen in the confusion matrix above in comparison to the actual target labels. Each row in the matrix represents an actual target label and each column represent a cluster. The matrix comprises five rows and five columns.

(d) What trends, if any, do you notice from the confusion matrix? Does it look like some clusters are able to pick up on a single label? When a cluster includes multiple labels, are they related?

The first row shows how reviews with score 1 are distributed across the 5 clusters. It is quite clear that most of the reviews with score 0 are part of cluster 0 and other clusters contain only small number of reviews with score. The second row is about neutral reviews and are distributed unevenly across all the clusters. The third row is about positive reviews and most of these reviews are clustered together in cluster 0. The reason that the last two rows are all zeroes is that there are only 3 target labels in the data after we relabeled the scores during data pre-processing. Therefore, no instances truly belong to these target labels. No, none of the clusters are able to single out any specific label as can be observed from the confusion matrix and examples shared above. Each cluster contains a mix of all labels and clusters are able to include reviews on similar topics. For example - the last cluster (cluster-4) contains multiple labels on dog and cat food.

Q3 - Comparing Classifiers

For the **next task**, the target was to develop models to classify the customer review automatically. The first two models are baseline models using Dummy Classifiers. Dummy Classifier ignores the input and predicts labels based on strategy. For one of the models, the strategy used is most_frequent. This strategy assigns the most frequent label (Score) to customer reviews. This strategy is generally used with imbalanced datasets. For our balanced dataset, this strategy will lead to poor f1, precision, and recall scores as highlighted in Table-2. Since it ignores the input data the accuracy of the model is extremely poor. Another strategy we used is stratified which assigns labels based on the empirical distribution of the labels. The accuracy of this model is the same as the previous but the precision improves slightly. Since both strategies yield similar results, it proves that our dataset is fairly balanced. These models are used as baseline models to compare against more complex models like Logistic Regression, SVM, Random Forest Classifier, etc.

Before developing the models, we need to vectorize the data. The two approaches used here are TF-IDF vectors and One Hot Encoding vector.

- **TF-IDF Vectorization** – It is a technique in NLP to convert raw text document into vector representation. This vector takes into account the Term Frequency(TF) and Inverse Document Frequency(IDF).
- **One Hot Encoding** – The count of each word in the document is used to form a vector but in this case each word is represented by a binary value of 0 or 1 based on whether word is absent or present in the document.

Since machine learning models cannot interpret humanly readable language directly, we need to convert our text into a numerical representation that can be used for machine learning or statistical analysis. Using scikit-learn library, we transform our training data and validation data into TF-IDF vectors using **TfidfVectorizer** and One Hot Encoding vector with **CountVectorizer**. With this data, we will train our machine learning models for predicting 'Scores' for customer reviews.

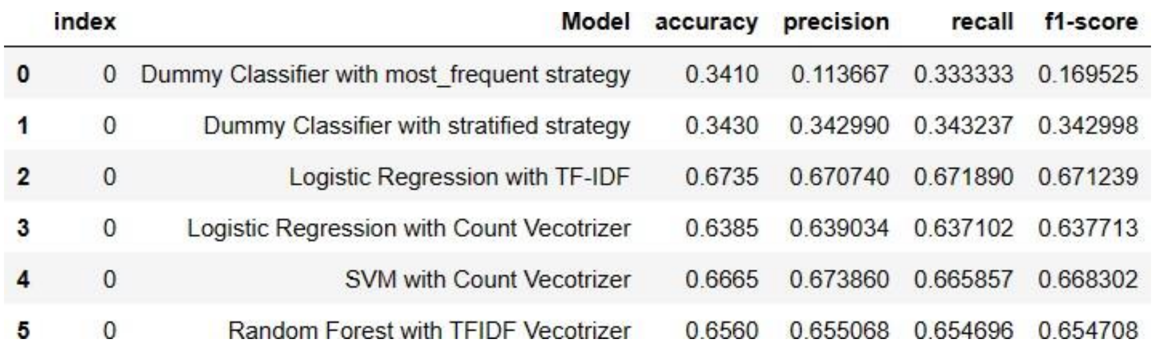
Using TF-IDF vectors we train a logistic regression model which gives an accuracy of about 67%. This model is trained on default parameters and can be tuned to achieve higher accuracy which will be discussed in a later section. Generally, logistic regression models are used for binary classification problems and use a sigmoid function to model the probability of an outcome. The evaluation metrics in Table-2 indicate moderate performance.

On the other hand, when a Logistic Regression model is trained using one-hot encoded vectors the accuracy drops to about 63%[Table-2]. The other evaluation metrics like f1, precision score, and recall score are also low in comparison to the previous Logistic Regression model. The reason for this degradation in performance can be attributed to the fact that the count vectorizer simply counts the occurrence of each word in a document, without considering its importance in the dataset. Whereas TF-IDF vectors represent the importance of each word in a document, based on its frequency and accommodate for rare words in the dataset. It emphasizes the importance of unique features in a document. That is why the Logistic Regression model when trained on TFIDF vectors gives better performance.

Support Vector Machine (SVM) is a state-of-the-art and powerful machine learning algorithm. When evaluating its performance on one hot encoded vector it is comparable to that of Logistic Regression on TFIDF vectors. One of the reasons is that SVM can handle non-linear decision boundaries effectively and can project data in higher dimensions. Therefore, the model achieved an accuracy of 66.6% even with one hot encoded vector.

Next, I selected Random Forest classifier as it can handle high dimensional data well. It is an ensemble method that uses the predictions from multiple decision trees to calculate the outcome. The tokenizer of choice was TFIDF vectorizer because of its ability to handle rare words and combine the two measures TF (Term Frequency) and IDF (Inverse Document Frequency). Using default parameters, the performance of this model is comparable to that of SVM and Logistic Regression model (with TFIDF vectors). The model achieved an accuracy of 65.6%. The f1, precision and recall score are approximately about 0.64.

The following table reports the accuracy, macro-averaged precision, recall, and F1 for all the classifiers [Table 2].



	index	Model	accuracy	precision	recall	f1-score
0	0	Dummy Classifier with most_frequent strategy	0.3410	0.113667	0.333333	0.169525
1	0	Dummy Classifier with stratified strategy	0.3430	0.342990	0.343237	0.342998
2	0	Logistic Regression with TF-IDF	0.6735	0.670740	0.671890	0.671239
3	0	Logistic Regression with Count Vecotrizer	0.6385	0.639034	0.637102	0.637713
4	0	SVM with Count Vecotrizer	0.6665	0.673860	0.665857	0.668302
5	0	Random Forest with TFIDF Vecotrizer	0.6560	0.655068	0.654696	0.654708

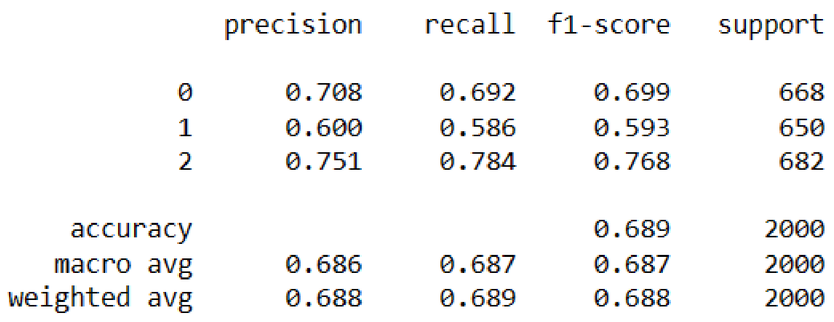
Table 2 – Evaluation metrics for baseline models

Q4 - Parameter Tuning

All the models in Table-2 are trained on default parameters. In this section we will use the Logistic Regression model and perform a grid search to tune the model using GridSearchCV and Pipeline implementation of scikit-learn. The pipeline consists of a TfidfVectorizer and a Logistic Regression model. The GridSearch function is used to perform a grid search. It takes hyperparameters, parameter grid and number of folds as input. The following parameters were used for grid search:

- **Penalty** – It adds penalty value to the function it is trying to minimize. This is done to prevent overfitting. The only possible values are L1, L2, ElasticNet and None. L1 penalty is also referred to as Lasso Regression which adds a penalty proportional to absolute value of coefficients. This reduces the coefficient values to zero and performs feature selection. L2 penalty is also called Ridge Regression which reduces the value of coefficient to near zero but not absolute zero. Elasticnet is a combination of both l1 and l2. On tuning, our model we the best paprameter was l2 penalty. This can be attributed to the fact that this dataset has large number of features, and a lot of the features are correlated. This is called 'the curse of dimensionality'. L2 regularization reduces the impact of this by adding a penalty. Overall L2 regression is a good choice for this dataset.
- **Sublinear TF** – For our model we have kept sublinear_tf parameter as True. This will apply logarithmic scaling to term frequency which will reduce the impact of frequently occurring words and emphasize on important and informative words.
- **Max Features** – The max features parameter is set to 7000 to reduce the dimensionality of the dataset. The grid search was performed in range of 5000 to 9000 with a step size of 1000. This parameter sets a trade-off between dimensionality and important features. Although the vocabulary size for our dataset is about 9000, we have set the max features value to 7000 so that we don't lose a lot of important information.
- **C** – This parameter defines the strength of the regularization. The possible range for this parameter is 10^{-3} to 10^{5}. A lower value of C corresponds to stronger regularization. GridSearch indicates that the optimal value of C for our use case is 1 which is also the default value of C.
- **Max DF** – This parameter ignores the words that are common across document. Grid search was performed between a range of 0.8 to 0.9 with a step size 0.1. The value set is 0.8 which ignores words that are common across 80% of documents. This effectively reduces the impact of common words. By ignoring words that occur in more than 80% of the documents, the resulting feature set may be more informative and focused on the key discriminative features in the dataset.

Based on the parameter discussed above, a logistic regression model was trained on tune tfidf vectors. The following image shows the classification report for the same.



```
              precision    recall  f1-score   support

           0      0.708     0.692     0.699       668
           1      0.600     0.586     0.593       650
           2      0.751     0.784     0.768       682

    accuracy                          0.689      2000
   macro avg      0.686     0.687     0.687      2000
weighted avg      0.688     0.689     0.688      2000
```

Image 3– Classification Report for tuned Logistic Regression

From the image above we can infer that the tuned Logistic Regression model performs better than the one trained using default parameters. The accuracy of the models increases from 67% to 68.9%. F1, Precision and Recall score also increase by approximately 0.2.

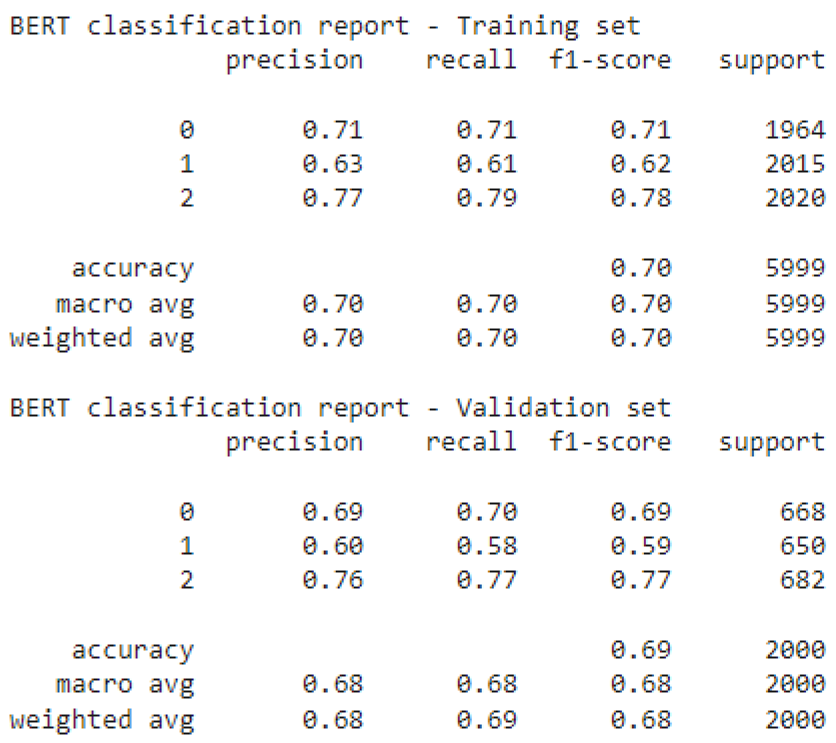
Q5 - Context vectors using BERT

(a) Encode the text of your documents using the 'feature-extraction' pipeline from the HuggingFace library with the 'roberta_base' model. Use only the first context vector for each document (which should represent the [CLS] token). Pass the context vectors (without any other previous features) into a Logistic Regression classifier from scikit-learn and train using the training set. Report the evaluation metrics on the validation set.

Feature extraction is process by which we can translate data into numerical representations which can be used by machine learning models to performs tasks and predictions. This is usually done by using pre-trained language models that have been trained on large amounts of text data and have learned to extract meaningful features from text.

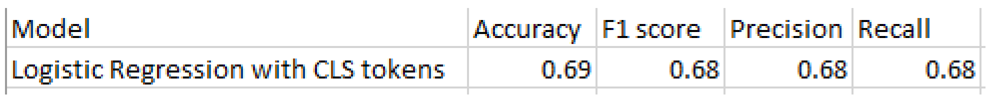
Hugging face is popular open-source library which provides pre-trained language models in NLP. The "roberta_base" model is one of the pre-trained language models available in the HuggingFace library. It is based on the RoBERTa architecture, which is a variant of the popular BERT model.

Using the Roberta tokenizer and feature extraction pipeline we will extract the first context vector of each customer review in the training dataset and pass it to Logistic Regression model along with its respective relabelled scores. The classification report below shows the performance of the model on training and validation dataset.



```
BERT classification report - Training set
              precision    recall  f1-score   support

           0       0.71      0.71      0.71      1964
           1       0.63      0.61      0.62      2015
           2       0.77      0.79      0.78      2020

    accuracy                           0.70      5999
   macro avg       0.70      0.70      0.70      5999
weighted avg       0.70      0.70      0.70      5999

BERT classification report - Validation set
              precision    recall  f1-score   support

           0       0.69      0.70      0.69       668
           1       0.60      0.58      0.59       650
           2       0.76      0.77      0.77       682

    accuracy                           0.69      2000
   macro avg       0.68      0.68      0.68      2000
weighted avg       0.68      0.69      0.68      2000
```

Image 3 – Classification Report Logistic Regression on Roberta Context vector



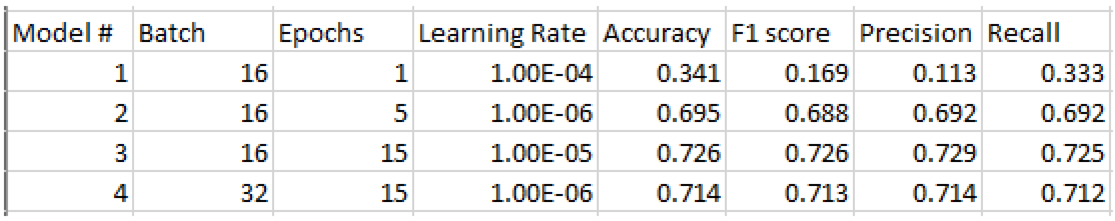
Model	Accuracy	F1 score	Precision	Recall
Logistic Regression with CLS tokens	0.69	0.68	0.68	0.68

Table 3 – Evaluation Metrics Logistic Regression on Roberta Context vector

(b) Train an end-to-end classifier using the 'trainer' function from the HuggingFace library, again using the 'roberta_base' model. Use a learning rate = 1e-4, epochs = 1, batch_size = 16 and no weight decay. Report the evaluation metrics on the validation set.

We will train an end-to-end text classifier using the "trainer" function from the HuggingFace library and the "roberta_base" model. We will be using a learning rate of 1e-4, a single epoch, a batch size of 16, and no weight decay. Generally, training a model with just 1 epoch is not enough because training a RoBERTa model requires several epochs (often tens or hundreds) to achieve good performance. The performace for this model is reported in Table-4

(c) Try different values for the model, learning_rate, epochs and batch_size. Normally, you would do some form of systematic search across these values, but due to computational costs, you should not do that. Pick three different sets of these hyperparameters and describe your motivation for these choices. Retrain the models from scratch on the training set and report the evaluation metrics on the validation set for those three settings in a table along with the hyperparameter settings from (b).



Model #	Batch	Epochs	Learning Rate	Accuracy	F1 score	Precision	Recall
1	16	1	1.00E-04	0.341	0.169	0.113	0.333
2	16	5	1.00E-06	0.695	0.688	0.692	0.692
3	16	15	1.00E-05	0.726	0.726	0.729	0.725
4	32	15	1.00E-06	0.714	0.713	0.714	0.712

Table 4 – Roberta Model

For Model-2 we kept the batch size same as Model-1 but we increased the number of epochs from 1 to 5 because it is not possible for Roberta model to learn patterns in the in just 1 epoch. The learning rate was kept at 1.00E-06 because larger learning rate could overshoot the optimal parameters or may take too long to converge. This change in hyperparameters had huge impact on the accuracy of the model as it increased from 34% to 69.5%.

For Model-3 we kept the batch size same and increased the number of epochs from 5 to 15 and increased the learning rate to judge the impact of learning rate on accuracy. This resulted in one of the best models in terms of accuracy.

For Model-4 we increase the batch rate to 32. The idea was that larger batch could help in optimization and help n generalization. The accuracy and other evaluation metrics reported by this model on validation set were very close Model-3 but still could not match its accuracy level.

Furthermore, experiments with this model could not be performed due to lack of computation power available. But in hindsight if we tweak the learning rate of Model-4 and increase it 1.00E-05 or 1.00E-04 it could have increased the accuracy as it is often recommended to increase the learning rate in proportion to batch size.

(d) Which performed best: the approach in part (a) using context vectors from the pipeline approach or using an end-to-end model in parts (b-c). How do these approaches differ? What is the likely reason for any performance difference?

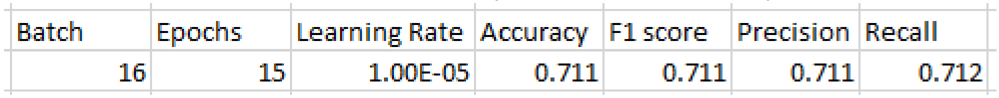
It is evident from Table 4 and Table 3 that using end to end classifier (part c) is better than using context vectors from pipeline. The given two approaches are different from each other because the latter uses pre-trained Roberta model for classification task whereas the former uses the pre-trained model only for feature extraction.

The latter is likely to perform better because it has an advantage of using a pre-trained model that can handle more complex and complicated scenarios based prior training while the latter (Logistic Regression model) had limited amount of training to handle such scenarios.

Q6 - Conclusions and Future Work

(a) Take the best approach from the prior questions (which should be trained on the training set) and evaluate it with the test set. Report the evaluation metrics. Provide a single confusion matrix of the classifications on the test set.

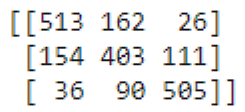
We will use Model-3 with 15 epochs, batch size of 16 and learning rate of 1.00E-05 to evaluate the test dataset since this model has the best accuracy. The table below reports the evaluation metrics:



Batch	Epochs	Learning Rate	Accuracy	F1 score	Precision	Recall
16	15	1.00E-05	0.711	0.711	0.711	0.712

Table 5 – Evaluation metrics on test data

The confusion metrics for test dataset is show below:



```
[[513 162  26]
 [154 403 111]
 [ 36  90 505]]
```

Image 4 – Confusion Matrix

(b) Manually examine the predictions on the test set. Analyse the results for patterns and trends using the confusion matrix. Hypothesise why common classification errors are made. Report on your error analysis process and summarise your findings. Be sure to give specific examples of the types of errors you observe.

As part of the error analysis process, I evaluated the model's performance on the test dataset, used confusion matrix as well as manually checked the predictions made by the model.

Now let us analyze the confusion matrix given above. The 3 x 3 matrix given above represents the three labels. It indicates the number of instances predicted to belong to a certain class/label (column) and actual class/label. We can summarize this matrix as:

- Row 1 – For Label-1, 513 instances were correctly predicted but 162 instances were classified as Label-2 and 26 as Label-3. Although the model is largely able to classify correctly but still there are quite a lot of misclassifications. Therefore, the performance of the model can be termed as moderate for Label-1.
- Row 2 - For Label-2, 403 instances were correctly predicted but 154 instances were classified as Label-1 and 111 as Label-3. It struggles in classifying the neutral comments largely because of the vague nature of such comments. These comments usually contain both positive and negative feedback and can be misclassified. Some of these comments can be even difficult for humans to classify correctly.
- Row 3 - For Label-3, 505 instances were correctly predicted but 36 instances were classified as Label-1 and 90 as Label-2. The confusion matrix is evidence that the system is able classify negative customer reviews easily.

The predictions made by the model were moderately correct (about 72%) as indicated by the accuracy levels of the model on the test dataset. The misclassifications can be attributed to the following reasons-

- **Out of vocabulary words** – Since the model is trained on a very small subset of the original data set it might not have encountered certain words before which were present test data but not in training data.
- **Misspelled words** – It was observed that a lot of the review had incorrect spellings, use of slang language. The misspelled words will also contribute towards OOV words. For example – 'cups' was misspelled as 'kups'.

(c) Discuss whether the final performance is sufficiently high for the stated purpose of this classifier. What impact would false positives and false negatives have on the deployment of this machine learning pipeline? Refer to the confusion matrix from (a) to support your claims.

The model can classify less 3/4th of the reviews correctly which is quite low. Similar systems deployed at commercial level have achieved high accuracy levels and are state-of-the-art models. We need to use hyper parameter tuning techniques to improve the accuracy of Roberta base models.

False positive and False negative can have a huge impact on the deployment of this machine learning pipeline because the error rate is still high when classifying Label 1 (score 0) and label-2 (score-1) as can be observed in row-1 and row-2 of the confusion matrix.

For example - In row-2 it can be seen that 154 neutral reviews are labelled as unsatisfactory review. If this automatically classified data is then used in a pipeline to feed another machine learning model then it can increase the error rate of the whole system to large extent. This clearly explains the impact False Negative or False Positive samples can have on the deployment this machine learning model.

(d) Could the deployment of this system have any negative societal effects?

Machine learning systems are highly dependent on the data used for training it. Such a system can have negative societal effects:

- **Bias** – Machine learning systems are prone to bias. These systems can often learn to prefer specific products. This means that the system will rate other products negatively and manufacturers can also incur losses in future.
- **Accountability** - The use of this system may lead to over-reliance on algorithmic decision-making, which could further lead to a lack of human intervention and accountability. If the system makes decisions based solely on data and without human intervention, it could lead to a lack of transparency and accountability in decision-making process.
- **Loss of jobs** - The use of this system may lead to job losses if businesses automate their decision-making processes. This could particularly attack jobs in marketing and retail industry.

(e) Propose further steps that could be taken to improve the classification effectiveness of the system.

The systems designed during this exercise are not ready for deployment. There are multiple steps that we can take to improve the accuracy and effectiveness of such systems:

- **Feature Engineering** – There may other features that can derived using the available data to improve the effectiveness of the system. For example – we can derive polarity scores for customer reviews and concatenate it along with tfidf vectors as an additional feature.
- **Hyperparameter tuning** – GridSearch is computationally expensive. We can use techniques like Bayesian Optimization or Gradient Based optimization which are computationally efficient. Using tuned parameter values for training models can potentially improve the accuracy of such systems.
- **Increase Training Data** – Currently, we are using only 5999 records to train the model. Increasing the amount of training data could help in improving the accuracy of the model

(f) How long did you spend on this coursework (in hours, an approximation is okay).

It took me more than 220 hours to complete this coursework.

Literature Review

Q7 - Research Paper Report

Enriching Word Vectors with Subword Information

Words with similar contexts tend to have similar meanings. Word embeddings are vector representations of words learned from a large dataset and capture the distributional properties. But traditionally, word embeddings treat each word as a basic entity and completely ignore the internal structure or morphology of the word.

The model proposed in the paper is an extension of the existing skipgram model (Mikolov et al. 2013b). Some other models like, a factored neural language model (Alexanderscu and Kirchhoff) also incorporate morphological information. Several earlier approaches, [2][3] used the concept of morphological decomposition. Cui et al. (2015) [4] proposed an idea where morphologically similar words were constrained to have similar representations.

The authors propose a method called as FastText which represents each word as a bag of character n-grams. The word embeddings are constructed by summing up the embeddings of all the n-grams that occur in the word. FastText allows the model to handle OOV words more effectively by learning embeddings for the subword units that make up the word. This also improves the embeddings of rare words. In addition to FastText, the paper also proposes a method for training word embeddings on multiple languages simultaneously, called vecMap. This method leverages the similarities between languages to learn cross-lingual embeddings that can be used for tasks such as machine translation and cross-lingual document classification.

The paper shows that FastText gives state-of-the-art performance when compared to existing word embedding methods on a range of benchmark tasks, particularly for morphologically rich languages. The performance of the model was then compared to skipgram and cbow by conducting 5 separate experiments using different datasets. For the correlation between human judgment and cosine similarity on the word similarity dataset, the FastText model outperformed all the baseline models except for the English WS353 dataset because it does not exploit subword information. For syntactic word analogy tasks, we observe a significant improvement, but for the semantic analogy tasks, it falls behind by a small margin. FastText model computes and constructs word embeddings for OOV words and thus, the effect of the size of the training dataset is reduced in comparison to the baseline models. The size of n-grams varies between 3-6 for this model which is enough to cover a wide range of information. Language modelling using RNN with 650 LSTM units also shows that vectors generated by this model lead to better perplexity scores. All this along with other qualitative analysis supports the author's claim that FastText is the superior model when compared to other methods that take subword information and even rely on morphological analysis.

FastText shows state-of-the-art performance without any pre-processing or supervision, but it incurs increased computational costs compared to traditional methods. This method also computes word embeddings for all the character n-grams, the model will be memory intensive and slower to train compared to other methods. As highlighted in the results as well, FastText is not well suited for applications where sub-word or morphological information is not useful.

References

1. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.
2. Ballesteros, M., Dyer, C. and Smith, N.A., 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. arXiv preprint arXiv:1508.00657.
3. Baroni, M. and Lenci, A., 2010. Distributional memory: A general framework for corpus-based semantics. Computational Linguistics, 36(4), pp.673-721.
4. Cui, Q., Gao, B., Bian, J., Qiu, S., Dai, H. and Liu, T.Y., 2015. KNET: A general framework for learning word embedding using morphological knowledge. ACM Transactions on Information Systems (TOIS), 34(1), pp.1-25.
5. Alexandrescu, A. and Kirchhoff, K., 2006, June. Factored neural language models. In Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers (pp. 1-4).
6. Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., 2017. Enriching word vectors with subword information. Transactions of the association for computational linguistics, 5, pp.135-146.