AML Assignment 1 : Project Report - Sristi Bafna

A. Preliminary details about the dataset:

The given dataset is a csv file containing 205053 entries under the following 6 columns:

1. Product
2. Product Price
3. Rate
4. Review
5. Summary
6. Sentiment

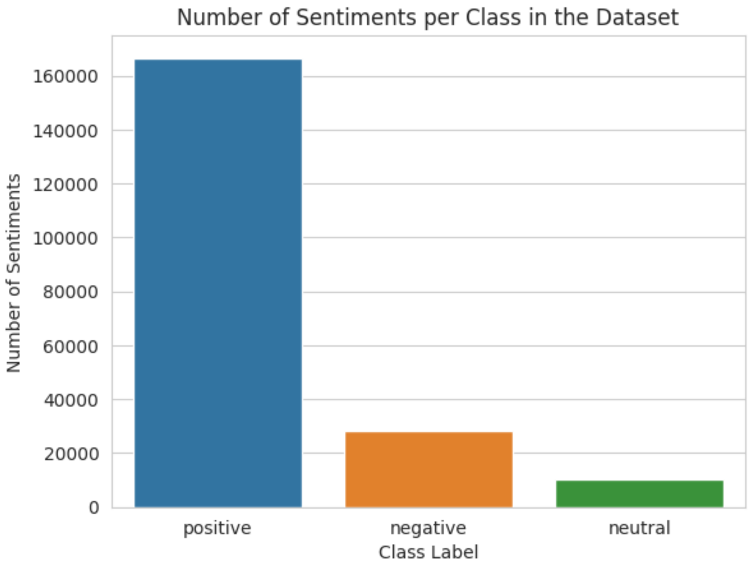
There are 958 products in the dataset.

B. Dataset Exploration:

Sentiment classes:

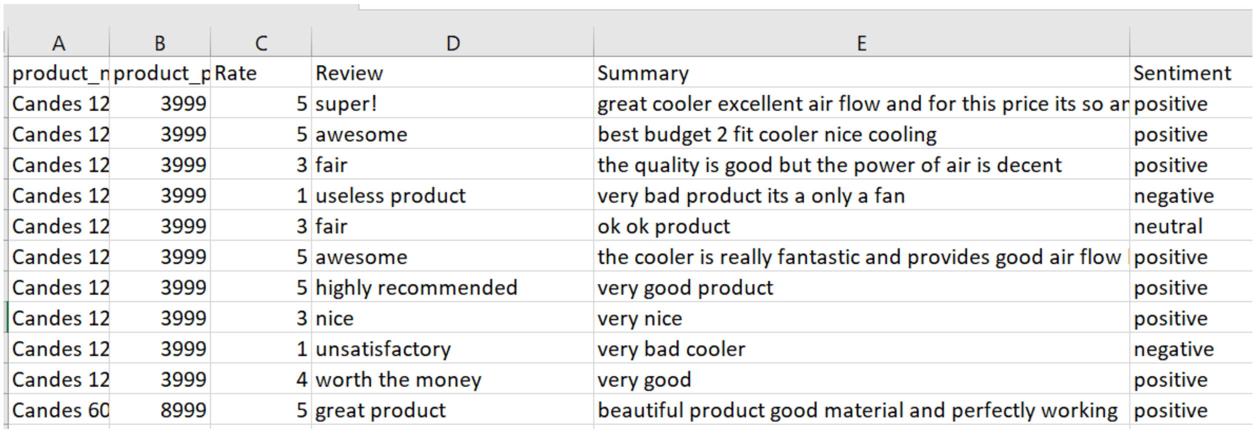
1. Positive
2. Negative
3. Neutral

The sentiment distribution was as follows:



1. As we can see, the dataset is highly imbalanced. Most reviews are classified for positive, while sentiments like neutral and negative barely have any occurrences.

2. As is visible, people have given reviews in phrases and not complete sentences. There are grammatical errors such as lacking apostrophes, unnecessary additions of articles and almost no punctuation. There are also a lot of typos throughout the dataset and a small typo could change up the sentiment of the whole review if context is not taken into account. For instance, “poor” was misspelt as “poar” and instead of “nice”, “ice” was written.



A	B	C	D	E	
product_n	product_p	Rate	Review	Summary	Sentiment
Candes 12	3999	5	super!	great cooler excellent air flow and for this price its so ar	positive
Candes 12	3999	5	awesome	best budget 2 fit cooler nice cooling	positive
Candes 12	3999	3	fair	the quality is good but the power of air is decent	positive
Candes 12	3999	1	useless product	very bad product its a only a fan	negative
Candes 12	3999	3	fair	ok ok product	neutral
Candes 12	3999	5	awesome	the cooler is really fantastic and provides good air flow	positive
Candes 12	3999	5	highly recommended	very good product	positive
Candes 12	3999	3	nice	very nice	positive
Candes 12	3999	1	unsatisfactory	very bad cooler	negative
Candes 12	3999	4	worth the money	very good	positive
Candes 60	8999	5	great product	beautiful product good material and perfectly working	positive

3. Another interesting thing in the above example itself is the use of phrases that are common lingo amongst humans but may be difficult for a machine to identify. For instance, “ok ok” is a very widely used phrase verbally but does not really have a proper connotation in core linguistics.
4. A lot of short forms have been used for instance, “awsm”,”aws” and “owsum” for “awesome”. The machine might find it difficult to relate this to a particular word or might completely misunderstand it.
5. A lot of the reviews are also unfinished which might mislead the sentiment. For example, some reviews mention just “nice to”.

We are using 1,64,032 data points in our training set and 41,009 data points in our test set.

C. Data Preprocessing and Augmentation

1. We use the NLTK library to preprocess the Review and Summary columns of the dataset. Preprocessing techniques used include the following:
    a. Removing punctuation characters
    b. Converting all the text to lowercase
    c. Tokenization - breaks down unstructured textual data into smaller units or tokens that can further be processed, analysed or used as input for our ML algorithms
    d. Stopword removal - process of filtering out common but unnecessary words in terms of sentiment classification like ‘a’, ‘an’, ‘the’, etc.
    e. Lemmatization - process of reducing words to their base or root form

    f. Join tokens back into sentences

2. We also perform 2 forms of conversion of textual data into numeric data:

    a. Converting the input into feature vectors
        i. Converting the input textual data into feature vectors using a specific feature extraction method (such as word embeddings, TF-IDF or Bag-of-Words)
        ii. The Sentiment column of the dataset is extracted as a numpy array and flattened
        iii. Data split into training (70%) and testing (30%) as specified.

    b. Converting the input into numeric sequences
        i. We use the Tokenizer class from Keras and keep the 5000 most important words in the vocabulary
        ii. Tokenize words and build word index based on their frequency
        iii. Convert the textual data now into sequences of word indices. Each word is replaced by its corresponding index in the word index built previously.
        iv. Unequal sequences are padded with 0s to make all of the same length.
        v. Labels for the dataset in the Sentiment column are one-hot encoded and then stored in a numpy array for further analysis

There are trade-offs for both of these methods.

The first feature vector method helps capture the semantic meaning of words and their relationships and can potentially result in more informative as well as rich representations of the input data. Word embeddings, for instance, can help represent words in a continuous vector space, facilitating capturing semantic similarities between words.

On the other hand, the second word index and one-hot encoding method is simpler and also more straightforward. It represents every single word with a unique integer which is an exceedingly simple input into any ML algorithm that accepts numerical inputs. It is more suited for simple models with no need to capture the semantic meaning of words and that are based more on frequency.

D. Performing Sentiment Classification

I performed sentiment classification using 4 different models/algorithms:

1. Using TextBlob: Accuracy = 83.3%

    a. I used TextBlob because of its built-in pretrained sentiment analysis model which would help me understand the baseline accuracy levels I should expect before training my own model.
    b. Additionally, TextBlob also has several additional text processing features such as part-of-speech tagging, spell checking and noun phrase extraction which are extremely useful for sentiment analysis.

2. Using Naive Bayesian Classification: Accuracy = 86.9%
    a. Although TextBlob essentially does Naive Bayesian classification, I decided to write code separately for this and the accuracy turned out to be quite similar as seen.
    b. I used Naive Bayes because it requires little training, has less computational overhead and can handle large datasets with high dimensional feature spaces.
    c. Naive Bayes is performed after the dataset has been trained for polarity and assigns each word a probability (likelihood) of having a particular sentiment.
3. Using Logistic Regression: Accuracy = 91.4%
    a. Even though LR only performs binary classification, I wanted to test the effectiveness of it as a sentiment analysis algorithm.
    b. LR is good for sentiment classification because it manages non-linear relationships between input and output variables and also works well with balanced datasets.

4. CNN using LSTM: Accuracy = 93%
    a. This was my main model for the assignment. It also achieved the highest accuracy of all.
    b. CNNs are good at detecting local patterns as well as features of a dataset. They use convolutional layers to automatically learn as well as capture relevant patterns (in our case, sentiment-bearing words or phrases).
    c. LSTMs are extremely effective at capturing long-term dependencies and sequential information in data. They maintain memory of previous inputs and thereby contribute to capturing contextual information.
    d. A combination of the 2 would thus be ideal for a sentiment analysis problem.

E. Our network

Here is a summary of the model:

I. Embedding
    A. This layer is responsible for converting the input text data into dense vectors of fixed size (128 here) to represent words or word sequences.

B. The input to this layer is a one-hot encoded representation of words with a vocabulary size of 5000.

C. The output of this layer is a dense vector representation of words that can capture semantic meaning and word relationships, which is important for NLP tasks like sentiment classification.

II. Spatial Dropout 1D

A. This layer applies regularization to the input data through dropout.

B. Dropout is essential to prevent overfitting because it randomly sets a fraction of input units to 0 during training, thereby encouraging the model to learn more robust features.

III. Conv 1D

A. This layer performs 1D convolution on the input data which is the process of applying filters or kernels to input data to extract local patterns or features.

B. In this case, the layer applies 64 filters with a kernel size of 3, which can capture local patterns or n-grams (3 consecutive words in this case) in the text data.

C. The activation function used is 'relu' (Rectified Linear Unit), which introduces non-linearity and helps in capturing complex patterns.

IV. Max Pooling

A. This layer performs 1D max pooling on the input data which is the process of reducing the dimensionality of the input data by selecting the maximum value from a group of adjacent values.

B. This can help to capture the most important features and reduce computational complexity.

V. LSTM

A. Long Short-Term Memory is a type of recurrent neural network (RNN) that can maintain memory of past inputs and capture contextual as well as sequential information.

B. The layer has 128 LSTM units, with a dropout rate of 0.3 during training and 0.2 during recurrent connections, which helps to prevent overfitting.

VI. Dense Layer

A. This layer is a fully connected layer that produces the output prediction probabilities for the three classes (3 in this case) using the softmax activation function.

B. Softmax activation function converts the output values into probabilities, summing up to 1, which represents the predicted probabilities for each class.

VII. Loss function

        A. We use the categorical cross entropy loss function which is the commonly used loss function for multi class classification tasks.

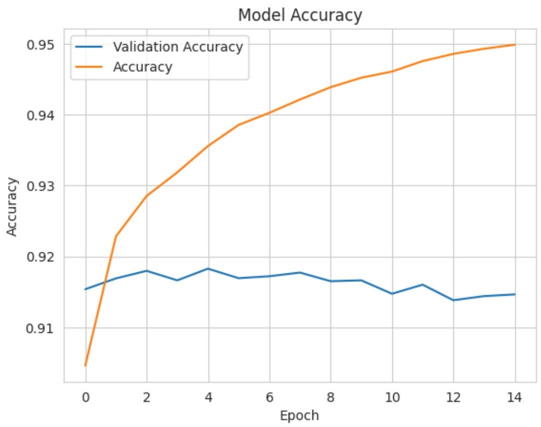
VIII. Optimization

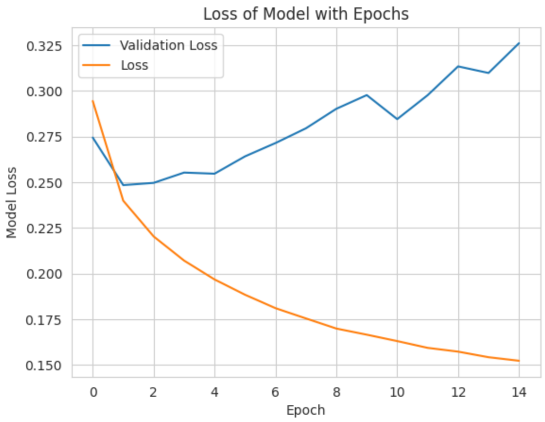
        A. We use the Adam Optimiser which is popularly used for NLP tasks.

IX. Finally, we evaluate our model based on the accuracy metric.

F. Results

1. **Base Experiment: TextBlob, 3-class Sentiment Classification**
    a. The input for this was directly the sentiments themselves. Baseline accuracy was established at 83.3%. Following this, I performed all preprocessing techniques to further increase my accuracy as well as altered the input for the algorithms to numerical instead of textual.
2. **Experiment 2: Naive Bayes, 3-class Sentiment Classification**
    a. Took feature vectors as input.
    b. With preprocessing it still gave just 86.9% accuracy indicating the need for a more complex model that would learn more intricacies in the dataset, especially things like contextual representation.
    c. Also indicated perhaps a different format of input for the dataset.
3. **Experiment 3: Logistic Regression, Binary Sentiment Classification (Positive/Negative)**
    a. Took feature vectors as input.
    b. With preprocessing it gave 91.4% accuracy but this was for binary classification, indicating the need for a more complex model for multiclass classification. Importance of contextual information learning re-highlighted here.
    c. Also indicated perhaps a different format of input for the dataset.
    d. Additionally, LR models do not perform well on a dataset with high class imbalance which was definitely the case for ours. This indicated the need for having some regularisation and normalization techniques to prevent a possible bias in the learning.
4. **Experiment 4: CNN+LSTM, 3-class Sentiment Classification**





a. Took sequential arrays as input.

b. With preprocessing it gave 93% accuracy for multiclass classification, thereby clearly being the best model for this task.

c. The CNN captured features well in the dataset and the LSTM helped with the contextual representation.

d. Additionally, dropout and max pooling layers helped with having generalised outputs which helped with the class imbalance.

e. The change in input format of the dataset may have also played a role in the better accuracy of this model.

f. We can see the model accuracy steadily increasing whereas the loss decreases initially but then fluctuates within a certain range. The loss decreases initially but then starts increasing. This could be due to the following reasons:

   i. Learning rate value : I would like to experiment with different learning rates to find the optimal value for the model.

   ii. Model Architecture : Possibly adding more blocks/layers to ensure better learning of the features because it is possible that the model architecture is too simple for a dataset of 200000+ data points and imbalanced classes which means the decision boundaries are also not too strong. Additionally, perhaps using more complicated architectures themselves like transformers could help with positional encoding alongside contextual encoding since LSTMs only capture context of previous words instead of the textual input as a whole.

   iii. Training data : The data at hand could be noisy implying that there is underlying variance and instability in the values given to the model itself which is why the learning is poor.

   iv. More data augmentation : Instead of lemmatization, stemming could be tried. Additionally, some NaN values seem to have persisted, their removal could be looked into.

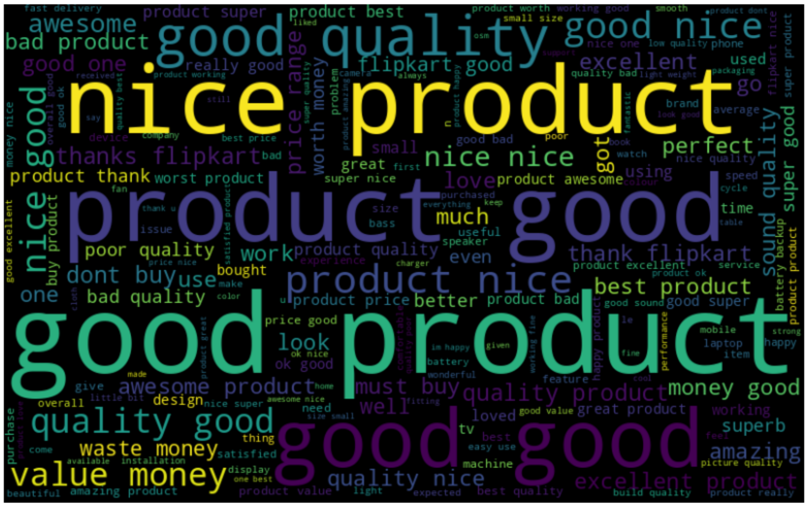
G. Task 3

K-means

1. Being an unsupervised machine learning algorithm, it does not require labelled data for training. In the case of identifying reasons for positive or negative reviews, we may not have labelled data that explicitly mentions the aspects or reasons associated with the sentiment.
2. It can be used for dimensionality reduction, where it can group summaries based on their similarity in terms of aspects or reasons mentioned. This can help in identifying common themes or patterns in the reviews that are associated with positive or negative sentiments. By reducing the dimensionality of the data, we can focus on the most relevant aspects or reasons that contribute to the sentiment, making it easier to identify top reasons for sentiments assigned.
3. It is also scalable, interpretable and quite flexible making a suitable algorithm for clustering reviews and summaries based on their sentiments.

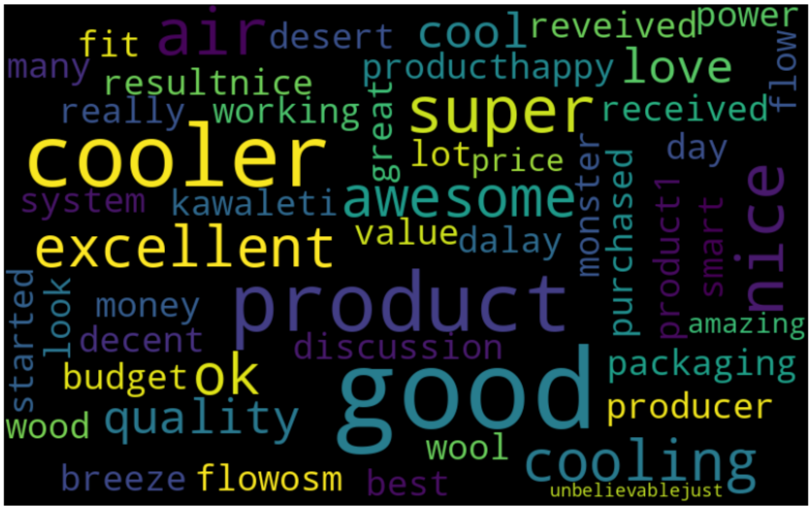
Here are word clouds for top 10 reasons for getting positive/negative/neutral sentiments/reviews for the given dataset:

1. Word cloud for the dataset:



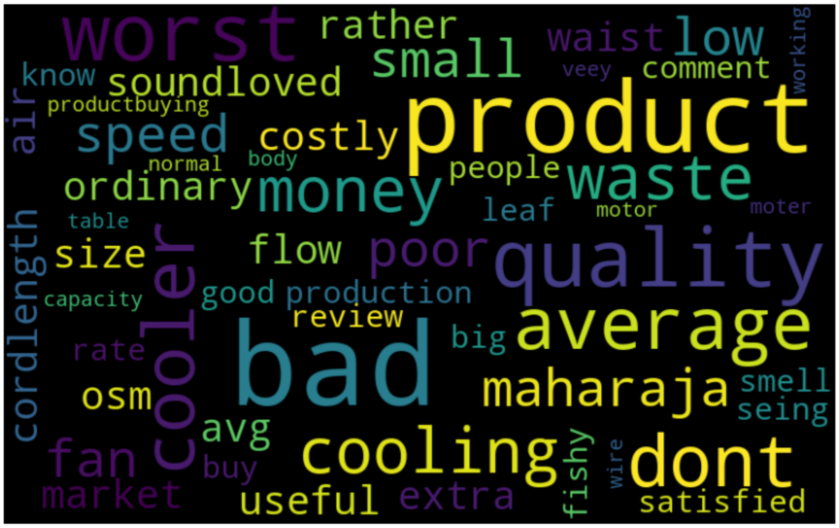
We can see that the word 'product', 'good', 'nice' and other fairly common words in our vocabulary dominated the reviews in the dataset.

2. Word cloud for the positive reviews:



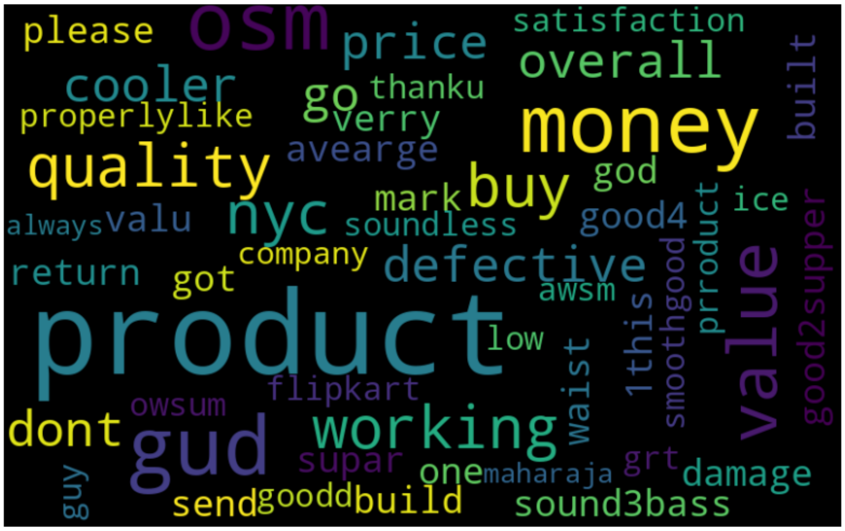
We can see that generally considered positive words like ‘awesome’, ‘good’, ‘excellent’ and others were common for this sentiment. Indicators used for reviewing like service, price and product function (cooling for the cooler) also appear a lot in conjunction with the adjectives.

3. Word cloud for the negative reviews:



Words like ‘costly’, ‘waste’, ‘bad’, ‘average’, etc. appear a lot here. Additionally, the model seems to have correlated the word ‘avg’ with ‘average’ due to the sentiments associated with it and perhaps words appearing with it in conjunction that may have the negative connotation.

4. Word cloud for the neutral reviews:



Mostly irrelevant words with no specific meaning appear here like ‘thanku’, ‘product’, and typos or common abbreviations that machines may not understand like ‘gud’, ‘osm’, etc.

H. Discussion and Action Points

Although the accuracy levels achieved (93%) were high, experimentation could be done with more complex model architecture, more data preprocessing and a lesser complex dataset to help with firmer decision boundaries for the classes. Future action points and experiments would involve exploring avenues mentioned as the possible causes for the drawbacks in the 4 experiments that were performed.