

Text Box

MSc in Business Analytics

Machine Learning and Content Analytics

*Assignment: Sentiment Analysis on Text Surveys*

|  |  |
| --- | --- |
| Authors: |  |
| Alvanou Marianna | p2822201 |
| Vlachakis Sotirios | p2822228 |

Table of Contents

[1. Introduction - Our Project – Goal 2](#_Toc145766229)

[2. Dataset Overview – Data Processing 2](#_Toc145766230)

[3. Methodology 5](#_Toc145766231)

[3.1 Feedforward Neural Network (FFNN) 5](#_Toc145766232)

[3.2 Sequential Model 9](#_Toc145766233)

[3.3 Greek Bert Model 12](#_Toc145766234)

[3.4 Topic Sentiment Analysis 13](#_Toc145766235)

[3.5 Clustering 15](#_Toc145766236)

[3.6 ABSA 18](#_Toc145766237)

[3.6.1 Rule-Based approach 18](#_Toc145766238)

[3.6.2 ABSA 20](#_Toc145766239)

[4. Tools 22](#_Toc145766240)

[5. Comments 22](#_Toc145766241)

[6. Members/Roles 22](#_Toc145766242)

[7. Time Plan 23](#_Toc145766243)

[8. Bibliography 23](#_Toc145766244)

# Introduction - Our Project – Goal

In this project, we asked to experiment with a real-world dataset, and to explore how machine learning algorithms can be used to find the patterns in data. Specifically, we wanted to know when a customer has a positive, negative, or neutral emotion through a text survey. The dataset is from the call center of a company that provides communication services and conducts daily surveys. We triggered to analyze such data due to the fact that we are working to the telecommunication company and we want as members of the reporting team at first to understand the customer, to know if the service of the call center is good or has problems – What are the problems, is it the fault of the company and each service that provide or something else – and then to find a way prevent negative comments but also to become better.

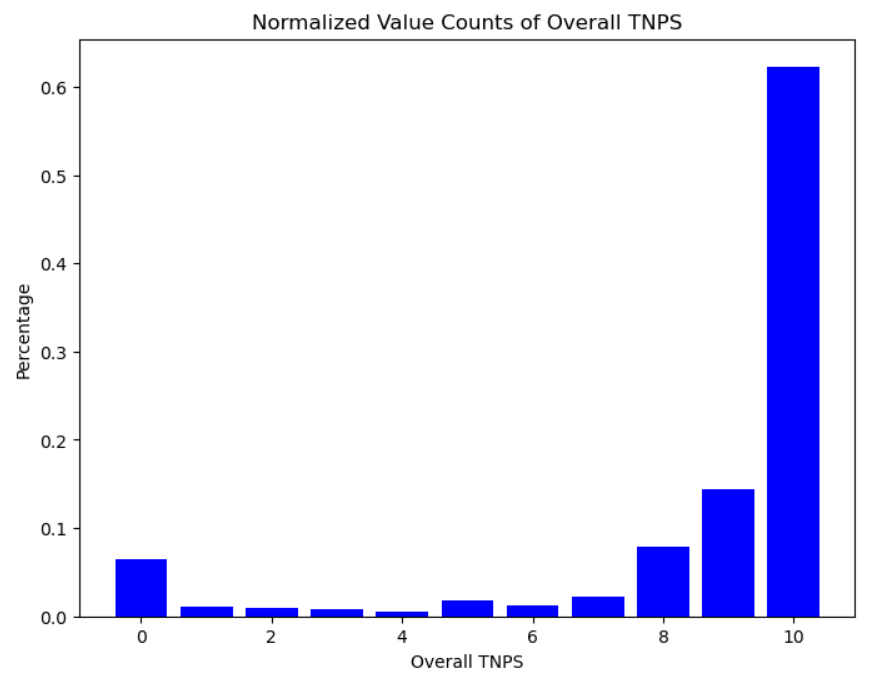
# Dataset Overview – Data Processing

Our dataset is an excel file that consists of 26065 surveys of May 2023. It has two columns, one for the score (Brand) and the other for survey (Comments) about the service offered to them by the company and the employee.

For the preparation of the models, it was necessary the data cleaning. We remove all the n/a values in both the Brand and the comment columns, we also create a new column (NPS\_Label) with the sentiment of the customer based on the score of the survey. The company has a counter to know which category each survey falls under. Surveys with Brand score between 9 and 10 have positive sentiment or in other words are promoters, scores between 7 to 8 are neutral and the last category is the detractors (Negative emotion) with score between 0 to 6.

We also have to change the font of the comments to lowercase due to future problems in the models. Our dataset is in Greek, so we create a mapping for the letters and implemented it in the comments.

Then we do Univariate Analysis to check the distribution of the score and emotion. As we can observe from the below bar plot, many customers in May 2023 were satisfied with their service after contacting the company's call center. Some of them were disappointed either by the service or generally with the company and a few were neutral.



*Figure 1: Distribution of the Brand Score*

The largest part of our dataset was positive surveys (76.7%), while Negative feelings emerged from only 13.1% of surveys.

A green circle with red triangle and yellow triangle

Description automatically generated

*Figure 2: Pie Chart of Sentiment*

Some of the words in the comments that show Positive emotion are “Αμεση εξυπητέρηση, ευγένεια, αμεσότητα” (Figure 3).

A close up of text

Description automatically generated

*Figure 3: Positive Words*

A close up of text

Description automatically generatedOn the other hand, words with negative feeling are “Πρόβλημα, εξυπηρέτηση, συμβόλαιο, σήμα, ιντερνετ”.

*Figure 4: Negative Words*

# Methodology

As it has been established, the problem is a multi-class text classification, and the following neural network techniques will be implemented to tackle the problem. Feet Forward Neural Network (FFNN), Sequential Model, Greek Bert Model.

We will also try a more advanced way of doing sentiment analysis and that is Aspect Based Sentiment Analysis (ABSA) which in a sense does sentiment analysis but for different topics and aspects of a text. We tried methodologies with LDA and clustering of our topics, we tried translating our comments and doing the previous methodology in hope of getting better results and we also tried a simpler rule-based ABSA method using NLP tools like stanza and NLTK.

Before we proceeded to the models, we needed to create a new column (NPS\_encoded), which is a coded column for the labels (emotions), this is because in all models it needed to be passed through encoding procedure.

Then we split the dataset into train, which is the biggest set (65%) - validation with 20% for the evaluation of the models and test set, the rest 15% for testing the data after the training.

## 3.1 Feedforward Neural Network (FFNN)

A diagram of a machine

Description automatically generatedA feed-forward neural network is a particular type of Multilayer Perceptron (MLP) in which the connections between the neurons do not produce cycles, allowing the data to flow just in one direction from the input layer via the hidden layers to the output layer (Figure 5). In a Feed Forward Neural Network, there are neither loops nor recurrent connections.

In this assignment we want to find the sentiment of customers when they make a survey. Are they satisfied with the service they received? Τheir problem is solved? What is the emotion they have after the phone call – chat with the employee of the company?

*Figure 5: FFNN Model*

As we start processing the data, after splitting the dataset into train – validation and test set, we should encode the dependent variable (y = NPS\_encoded) with the help of One-Hot-Encoder method and independent (x = Comments) variable through Count Vectorizer method.

More specifically, One-Hot-Encoder is for encoding categorical features as a one-hot numeric array. Each category (Positive, Neutral, Negative) is represented by zero (0) or one (1) in each comment / survey. For instance, if the score of the survey is Positive then in that place the 1 is shown and in the other will have 0.

The Count Vectorizer transforms a group of text documents into a matrix with each row representing a document and each column indicating a word from the vocabulary. The matrix entries show how many times each word appears in each document. Count Vectorizer removes from the vocabulary accents and diacritics using Unicode normalization, is also removes stopwords and limits the vocabulary size to the top max\_words = 500 most frequent words.

The model is set up with two dropout layers and two hidden dense layers, each of which has 512 units and a ReLu Activation Function that introduces non-linearity into the network. Dropout is a regularization method that deactivates a portion of neurons at random during training to avoid overfitting. The final layer then includes a softmax activation function. Softmax is appropriate for multi-class classification issues since it transforms the model's output into a probability distribution over the classes.

A screenshot of a computer program

Description automatically generated

*Figure 6: FFNN Model on text surveys*

To configure the learning process of the neural network model we use model.compile with loss = ‘Categorical\_crossentropy’, which measures the difference between the predicted class probabilities and the actual one-hot encoded target values. The optimizer parameter specifies the optimization algorithm used to update the model's weights during training and especially ‘Adam’ that handles weak trends and accelerates convergence by changing the learning rate for each parameter based on previous gradients. Finally, the evaluation metric that we use for this model is ‘Accuracy’, which calculates the accuracy of the model’s predictions compared to the actual value.

After this process we fit the model, but before that we create an early stopping callback – if it doesn’t improve for 2 consecutive epochs, the training will stop early. We also combine early stopping with the Checkpoint (save the model's weights during training if the validation accuracy improves) and the number of epochs is 10. We checked with different dense layers, dropout rates, with or without weights, callbacks and early stopping but the best model regarding the accuracy and the confusion matrix was the above.

The accuracy of the model is reached at 69.457 % and the categorical\_crossentropy: 74.752 %. As we can understand from the above pictures (Figure 7), after the 2nd epoch the model does not train any more, as the validation loss (green line) keeps increasing and in parallel the validation accuracy starts to decrease after 2nd epoch.

A graph with a line and a green line

Description automatically generated with medium confidenceA graph of a graph with a line

Description automatically generated with medium confidence

*Figure 7: Training and Validation Loss - Training and Validation Accuracy*

Confusion Matrix is also a key to our analysis. The True Positive values are diagonal in the table, which means that 2074 surveys are predicted with the true sentiment. The model seems to predict wrong those which are Positive, but it predicted as Neutral (359) and some of them Negative (251).

A screenshot of a graph

Description automatically generated

*Figure 8: Confusion Matrix*

Τhis can also be noticed from the Area Under Curve (AUC), and measures how well the classified the predictions. Negative (class 2: magenta) and Positive emotions (class 0: navy) are predicted quite good, while neutral no, because the area under the curve (orange) is very small and close to the diagonal, something that means that neutral feeling drives into wrong decisions.

However, the AUC = 0.7928, in comparison True Positive and False Negative, which is an acceptable score for the choice of the particular model.

A graph of a line graph

Description automatically generated with medium confidence

*Figure 8: ROC*

A screenshot of a graph

Description automatically generatedThe last step of this model analysis is the Classification Report (Figure 9), a total view of our Feedforward Neural Network Model. Recall defines that 73% of comments/ surveys are found Positive 74% Negative respectively against the total number and 35% found Neutral. The precision of the Positive Sentiment has the highest score 92%, support metric is the true value of each sentiment.

*Figure 9:* Classification Report

## 3.2 Sequential Model

Sequential Model is also a specific type of Multilayer Perceptron (MLP), is a linear stack of layers where each layer has connections to the previous and next layer. A particular implementation of a feedforward neural network offered by well-known deep learning frameworks like Keras. It is referred to as "sequential" because it is simple to construct a stack of layers that is linear and in which the output of one layer serves as the input for the following layer.

Here we split the dataset into a train set, which is 80% of the data size and test the rest 20% of the data. We similarly to the previous model, calculate the class weight (based on the distribution of distinct classes in the training labels, it balances the class weights).

Next steps are to transform the text data into a format that can be fed into a neural network for training and evaluation. Set the vocabulary size = 15000, including the most frequent words in the dataset and the maximum length of the sequences = 120, in order all sequences have the same length for consistent input to the model. Then, we tokenize the comments into integers, with the maximum number of words to keep in the vocabulary based on the word frequency, 15000 the same size as the vocab\_size to avoid any inconsistencies. The tokenizer learns the mapping between words and numbers by fitting on the training data, and it develops a vocabulary using the words in the training set. Furthermore, the training and test data (training\_reviews, testing\_reviews) are transformed into sequences of numbers, where each integer represents the index of a word inside the tokenizer's lexicon. After this process we should ensure that the sequences both in train and test sets have the same length. To do this, we should pad any sequences that are lower than max\_length with zeros, while trimming those that are greater. The integer-encoded labels are transformed into one-hot encoded format using the “to\_categorical” function of TensorFlow's Keras utilities. Each label is represented by way of one-hot encoding as a binary vector with just one element (1, indicating the class) and all other elements (0).

The optimal model is saved based on validation accuracy using a ModelCheckpoint callback (early stopping and the best weights of the model) after the model has been built with the proper metrics, loss (categorical\_crossentropy), and optimizer (Adam). The Sequential Model for multi-class classification uses an embedding layer to process the input text data (Comments), followed by a flattening layer, passing the text through a hidden dense layer activated by ReLu, and then applying a softmax activation in the output layer to provide class probabilities.

A screenshot of a computer program

Description automatically generated

*Figure 10:* Sequential Model

The model accurately predicted the sentiment of around 74.749% of the reviews in the test set. The degree to which predicted probabilities match the actual labels is shown by the categorical cross-entropy loss, which is 78.158%. As we can observe from the below pictures (Figure 11), a significant increase in val\_accuracy is observed in this epoch 3, reaching 0.7425, in the next epoch drops indicating a reduction in performance. Since there was no improvement for two consecutive epochs (4th and 5th) the model stops training due to early stopping. However, a key indication is also the val\_loss metric that decreased rapidly until the 4th epoch and after that started to fall more stably.

A graph with a line and numbers

Description automatically generatedA graph with a line and a line

Description automatically generated with medium confidence

Figure 11: *Training and Validation Loss - Training and Validation Accuracy*

The Area Under Curve of 0.7529 indicates that the model seems to be better than random, but it may have trouble differentiating between classes, maybe as a result of overlapping or unbalanced data. The Class 0 of Positive and Class 2 of Negative are predicted with high probability 82% and 85% respectively, while Neutral give the impression of faulty results.

A graph of a number of different colored lines

Description automatically generated

*Figure 12: ROC*

The Confusion Matrix shows us that Neutral feelings are predicted false as Positive (241), some of the Positive emotion comments are predicted as Neutral (173). The True Positive values of this matrix are 2232, while the False Negative are 754.

A blue squares with numbers

Description automatically generated

*Figure 13: Confusion Matrix*

## 3.3 Greek Bert Model

Greek Bert is a Transformer-based language model specifically trained for the Greek language. Bert is an innovative language model that has completely changed several natural language processing (NLP) activities. By taking into account both left and right context, this sort of transformer model is able to accurately represent the context of words inside a phrase.

A computer screen shot of a keyboard

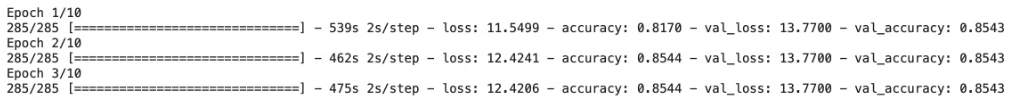
Description automatically generated

*Figure 14: Greek Bert*

In this model we use only 2 out of 3 sentiments, Positive and Negative. First of all, we load the pre-trained Bert tokenizer and the model for the Greek language. Comments/ surveys are preprocessed and tokenized using the tokenizer before being fed into the Bert model. We split the data for training, validation, and testing and then convert the labels/ sentiment into to One-Hot encoded format (numeric values 0 and 1).

Finally, the Adam optimizer and categorical cross-entropy loss are used to build the BERT model. The training data is utilized to train the model, and early stopping is done depending on validation accuracy.

As we can observe from the following figure 15, the model ends at the 3rd epoch, but we have the same validation accuracy and validation loss something that shows us that something goes wrong. Maybe due to the fact that the dataset is unbalanced.



*Figure 15: Greek Bert - epochs*

## Topic Sentiment Analysis

In this methodology we initially tried to tackle the way of ABSA by finding topics in our comments and then labeling the sentiment of these topics. After the initial preprocessing, we also tokenized and vectorized our comments. Latent Dirichlet Allocation (LDA) is an unsupervised machine learning algorithm than can discover latent topics in a corpus of text and we used that in order to “categorize” our comments into three different groups. These groups were created by utilizing LDA which selected the top 10 most common words for each group. By observing these words below

A close-up of a white background

Description automatically generated

we tried to identify what these topics could be and concluded into these:  
  
"Service Quality & Staff Interactions", "Technical Issues & Brand Interactions:", "Service Excellence & Problem Resolution".

Diving deeper to these topics we discovered the underlying emotions for each one.

A screenshot of a computer

Description automatically generated

A graph with green and blue bars

Description automatically generated

We can easily observe that the “Positive” emotional is dominating at our comments and most of them have been assigned to the “Service Quality & Staff Interactions” topic.

This methodology seemed like it couldn’t classify our comments with precision and that is why we will move to a different grouping method.

## Clustering

The different approach for grouping our comments is clustering, and we are going to use two different algorithms, K-means and DBSCAN.

Firstly, we used Greek-Bert in order to generate embeddings for our comments which represent individual words as dense vectors such that semantically similar words are mapped to nearby points in the embedding space.

Then by implementing the “elbow” method we tried to find the optimal number of clusters.

A graph with a blue line

Description automatically generated

As seen at the graph above, the optimal number seems to be 2. The results of K-means with two clusters are shown below, with the help of t-sna which is a dimensionality reduction method.

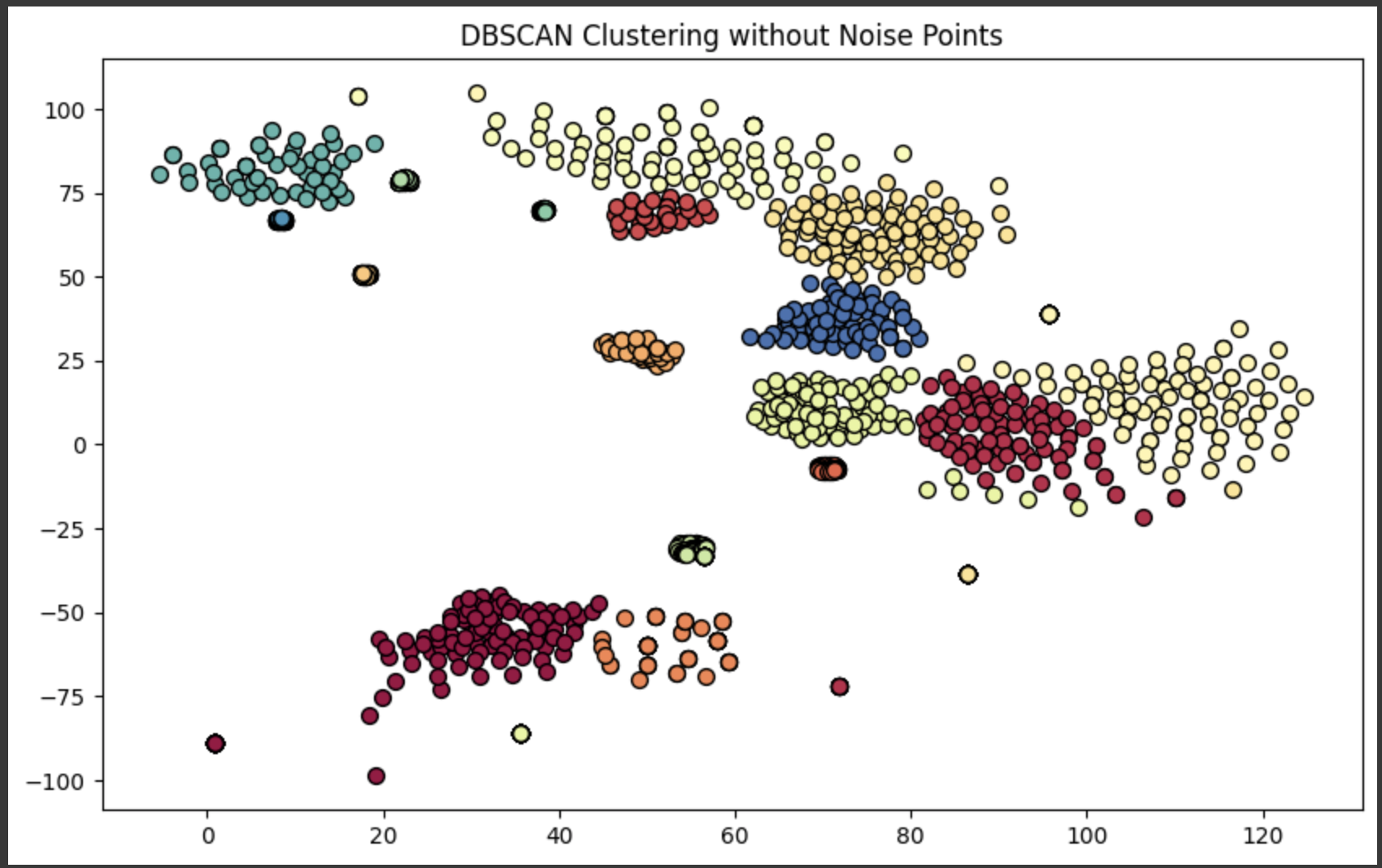
A red and grey blotches

Description automatically generated

The results seem a bit strangled, a fact that makes us think that there is no easy way of grouping our comments. But we can distinguish some groups so we will try a frequency-based method, DBSCAN.

A black and yellow dot diagram

Description automatically generated with medium confidence



By using DBSCAN we managed to identify 17 different clusters which seem like words very similar to each other, but all in all the noise (black points) is very high. Also, these 17 clusters created are clusters with the exact same comments as seen below.

A screenshot of a computer screen

Description automatically generatedThat tells us that our efforts of clustering our texts isn’t achieving anything, and that’s because of the nature of our data. They are customer’s comments which can vary a lot both at grammar and vocabulary, having different meanings and Greek language by itself is challenging for these kinds of tasks.

\*We had exactly the same results when we tried to translate our comments to English.

A diagram of a cluster of dots

Description automatically generated

## ABSA

### Rule-Based approach

Since we translated our comments, we tried to implement a more basic ABSA method based on rules. We lowercased and tokenized our comments and then for each sentence we performed POS Tagging. POS Tagging is the process of assigning each word in a sentence its corresponding part of speech, such as noun, verb, adjective, based on its definition and context and combining Dependency Parsing in order to determine the grammatical structure of the sentence, we were able to identify our aspects. In our case, as aspects we consider the nouns of each sentence and adjectives or adverbs usually provide sentiment information. If we find a noun next to another noun, we consider them as one and we unify them.

A diagram of a good but the good but the good

Description automatically generated

A diagram of a process

Description automatically generated

Next, we create a sample dataset in order to test our algorithm. Our dataset contained sentences like:

"The Sound Quality is great, but the battery life is very bad."

"The display is beautiful, but it overheats often.",

"Performance is exceptional!"

And the results were these:

A screenshot of a computer

Description automatically generated

As we can observe in the Analysis Result section, our model understood that “Sound Quality” (unified as we stated earlier) and “battery life” (again unified) for the first sentence are aspects, and “great”, ”bad” are their corresponding words of showing some positive or negative emotion.

Consequently, because of the fact that we only got labels (emotions) for the whole comment, we will try to extract emotions for each one of our aspects using pretrained model VADER.

For our sample sentence "The Sound Quality is great, but the battery life is very bad." we can see at the output this exact fact, that for example, “Sound Quality” has a positive emotion and “Battery life” has a negative emotion.

A screenshot of a computer

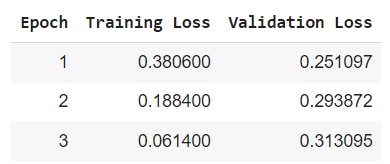
Description automatically generated

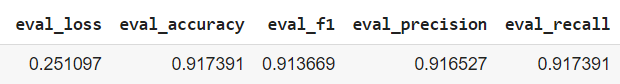
### 3.6.2 ABSA

Afterwards, we tried to cluster our aspects, by first vectorizing using word2vec, and Kmeans for clustering. Initially we chose 5 clusters, but as we see at our plot, after performing dimensionality reduction, we cannot observe clearly defined clusters.

For our next step, we will try and create a machine learning model in order to try to train it based on our data which we now have labeled, (every aspect has a sentiment label).

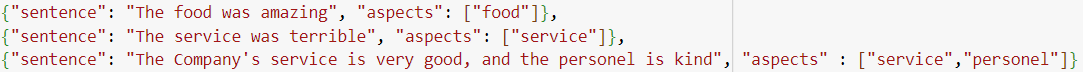
We used BERT, which is a transformer-based model neural network model. In the process of training further Bert on our data, we saw the below results.





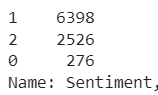
As we were advancing at epochs, the training loss was reducing but the validation loss was becoming bigger. That’s a sign of overfitting and that is why we kept the model trained until epoch 1.

Then we tried to test our model with a sample dataset,



we observed that our model seems to work fine for the sentences with one aspect



Where 2 is ‘positive’ and 0 means ‘negative’, but for sentences with 2 aspects both sentiments were neutral, something that can be because of the imbalance in our dataset.

That is why we will compute the balanced accuracy for our dataset:

# Tools

We use Oracle SQL to extract the dataset, Jupiter Notebooks and Google Colab in order to create the models for our analysis.

# Comments

|  |  |  |
| --- | --- | --- |
|  | True Positive | False Negative |
| FFNN | 2074 | 942 |
| Sequential | 2232 | 754 |

Greek Bert is not a good model for this analysis or in other words the dataset is not convenient to use this model.

+++++

Our future work will be to test with more data from different time points in order to see the models' predictions and how well or not they perform. Then we'll collect surveys from a different company and apply the models.

# Members/Roles

The team members of this assignment are: Alvanou Marianna and Vlachakis Sotiris. We both studied Statistics and Insurance Science at the University of Piraeus, we were coworkers in a Telecommunication Company, and we also collaborated many times in the past on both University assignments and work projects. These are some of the keys that push us to work together for this task.

The roles of each member are:

**Alvanou Marianna (p2822201):**

Discuss the Business Case, extract the data, data cleaning, create some models, write the report and presentation.

**Vlachakis Sotiris (p2822228):**

Communication with the professor, discuss the business case, data cleaning, create some models, write the report and presentation.

## Time Plan



# Bibliography

* [Feedforward neural network - Wikipedia](https://en.wikipedia.org/wiki/Feedforward_neural_network)
* [NDS1 (wiley.com)](https://media.wiley.com/product_data/excerpt/19/04713491/0471349119.pdf)
* <https://wiki.pathmind.com/neural-network>
* <https://wiki.pathmind.com/multilayer-perceptron>
* [machine learning - Why ReLU is better than the other activation functions - Data Science Stack Exchange](https://datascience.stackexchange.com/questions/23493/why-relu-is-better-than-the-other-activation-functions)
* <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>
* [How to Save and Load Your Keras Deep Learning Model - MachineLearningMastery.com](https://machinelearningmastery.com/save-load-keras-deep-learning-models/)
* [Practical Text Classification with Python and Keras – Real Python](https://realpython.com/python-keras-text-classification/)
* [Introduction to Machine Learning, Neural Networks, and Deep Learning - PMC (nih.gov)](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7347027/)
* <https://monkeylearn.com/sentiment-analysis/>
* <https://huggingface.co/nlpaueb/bert-base-greek-uncased-v1>
* Haris Papageorgiou - Slides
* Georgios Perakis – Lab sessions