C O V E N T R Y
U N I V E R S I T Y

Faculty of Engineering, Environment, and Computing

School of Computing, Electronics, and Mathematics

Master of Science in Data Science and Computational Intelligence

7151 CEM- Computing Individual Research Project

Predicting Airline Passenger Satisfaction Using Online Reviews

Author: Anjanajayanthi Sankararamanujam

SID: 12714306

Supervisor: Prof Reda Al Bodour

Submitted in partial fulfillment of the requirements for the Degree of Master of Science in Data Science and Computational Intelligence

Academic Year: 2022/23

Declaration of Originality

Statement of copyright

Statement of ethical engagement

Signed: Anjanajayanthi Sankararamanujam Date:10-04-2023

Please complete all fields.

First Name:	Anjanajayanthi
Last Name:	Sankararamanujam
Student ID number	12714306
Ethics Application Number	P148735
1st Supervisor Name	Prof Reda Al Bodour
2nd Supervisor Name	Prof Rochelle Sasman

Abstract

Airline passenger satisfaction is a crucial component of the airline sector because it directly impacts both the customer experience and the airline's reputation. In the past, airlines have gathered input on customer satisfaction through surveys. Customers are increasingly adopting online forums to voice their opinions about airline services, though, as internet and social media use grows. As a result, studying online reviews is turning into a crucial tool for figuring out customer contentment and enhancing airline services. The study on leveraging online reviews to predict airline passenger satisfaction is presented in this report. The purpose of the study is to assess how well a Bidirectional Encoder Representations from Transformers (BERT) model performs in categorising online evaluations into three groups: positive, negative, and neutral. The 1097 reviews that make up the dataset for the study were taken from OpenML, a public source for machine learning datasets.

A cutting-edge natural language processing approach called BERT has been demonstrated to be successful in a variety of language tasks, including sentiment analysis. The model is a deep neural network architecture that processes the input text and produces context-sensitive word embeddings using a bidirectional transformer encoder. The model can be fine-tuned for certain tasks like sentiment analysis because it has already been pre-trained on a big corpus of text data. The comments have been classified as positive, negative, or neutral and are presented as text data. With 70% of the data utilised for training and 10% each for validation and testing, the dataset was randomly divided into training, validation, and testing sets. The Hugging Face Transformers library's BERT model was utilised for classification. The training data were used to fine-tune the model, and the testing data were used to evaluate it. Precision, recall, F1-score, and accuracy were employed as evaluation metrics in the study, along with a confusion matrix.

According to the study's findings, the BERT model scored 77% on the testing data's weighted F1 score. The model's high levels of precision, recall, and accuracy also suggest that it was successful in predicting passenger satisfaction based on internet reviews. According to the confusion matrix, the model was most accurate at predicting negative ratings, which were then followed by positive and neutral reviews.

Keywords- Airline passenger satisfaction, Text Classification, Online Reviews, User-Generated Content, Sentiment Analysis, BERT, Transformers, Natural Language Processing.

Table of Contents

Acknowledgements

I want to sincerely thank everyone who helped me finish this report and for their support throughout the process. First of all, I want to express my gratitude to my supervisor for the essential advice and criticism he gave me during the project.

I am also appreciative that Coventry University gave me access to the resources and knowledge I needed.

I owe my colleagues my gratitude for their thoughtful criticism and remarks on my work.

Last but not least, I'd like to express my gratitude to my family and close friends for their constant support and inspiration during this endeavour.

1 Introduction

Client satisfaction is a key goal for any business, particularly commercial ones. It is a statistic used by business owners and marketers to determine how effectively their services are operating from the perspective of customers. It may also reveal whether or not the customers are likely to return frequently. Also, it can indicate that your clients are dissatisfied and might be on the verge of leaving using your services. Because of this, measuring customer satisfaction level gives businesses incredibly important information about whether they are heading in the right path or not. This principle applies to the aviation sector as well (Liu et al.,2022).

The fourth year of battling the coronavirus pandemic is about to begin for airlines. In some nations, the limitations and difficulties still exist, and the rules and regulations of the government are still in force. However, we can see that limits are remarkably easier now than they were in 2020. The difficulties, however, have persisted despite everything, from new viral variations to changing government regulations regarding testing and travel restrictions. Airlines' ability to design and carry out predictable schedules as well as the effect on customer confidence have had a significant negative influence on revenues and budgets. Due to these chaotic circumstances, fewer people travel, and global airline competition is at an all-time high. Customer happiness is very crucial, especially at this time. It is essential to predict what will satisfy customers and how to win them back.

Passenger satisfaction is a difficult thing to achieve in the aviation sector. When comparing the aviation industry to other sectors, it is clear that much effort is needed to meet the goal of customer happiness. The service begins with the airport of departure, its amenities, the planned timing of the journey, and whether or not it is convenient (Abualsaud et al.,2021). Other factors to consider include online and mobile boarding, baggage management, check-in, in-flight amenities, and arrival or departure delays. The situation is really crucial, and it requires innovative methods to handle employing cutting-edge technologies.

1.1 Background to the Project

Millions of people use aeroplanes as a form of transportation every day, making it one of the most popular. In the aviation sector, passenger satisfaction is crucial since it affects airlines' reputations, customer loyalty, and financial performance. Although airlines work hard to deliver exceptional customer service, a number of factors, such as luggage handling, in-flight amenities, and flight delays, can have an impact on how satisfied passengers are (Liu et al.,2022). As a result, pinpointing the variables that influence it and enhancing the customer experience depend on precisely anticipating passenger happiness. The prediction of airline passenger pleasure has been studied using a variety of methods, including machine learning, sentiment analysis, and natural language processing techniques. Some of these methods, however, have drawbacks, including the inability to handle huge datasets and the difficulties in handling unstructured text data.

A cutting-edge language model called BERT (Bidirectional Encoder Representations from Transformers) has been demonstrated to perform better than previous models in a number of natural language processing tasks (Gupta et al.,2022). A neural network-based technique called BERT creates word embeddings that accurately capture the meaning of words in context by learning the contextual relationships between the words in a phrase. Text classification, sentiment analysis, and question-answering are just a few of the areas where BERT has been successfully used (Kumar et al.,2022). In order to reliably forecast airline passenger pleasure, BERT model has been employed in this study. Our goal is to create a BERT-based model that can assess customer reviews and feedback, pinpoint the elements that influence satisfaction, and reliably forecast customer satisfaction scores. By utilising the strength of BERT in processing huge, unstructured datasets and identifying contextual links between words, we hope to solve the shortcomings of present techniques.

This initiative has great potential relevance and influence. Airlines can pinpoint areas for improvement, modify their offerings to better suit customers' wants, and improve the entire passenger experience by properly anticipating passenger satisfaction. The results of this study can influence airline policies and procedures and help the aviation sector become more competitive and sustainable.

1.2 Research Questions

This initiative aims to assist airlines in achieving the target level of customer satisfaction by enabling them to concentrate on the most crucial elements that may also be the elements that customers find most appealing. Its primary objective is to respond to the questions listed below:

- Estimating a customer's level of satisfaction with the entire service offered by the airlines.
- Are internet reviews more accurate and useful for predicting airline passenger pleasure than other information sources like surveys?
- Is it possible to train the BERT model, a language-based machine learning model, to recognise problems that can signal passenger unhappiness (e.g., baggage handling, delays, etc.)?

1.3 Project Objectives

The aim of this work is to develop a predictive tool utilising the BERT model that can help airline firms enhance their customer experience by analysing online customer reviews and accurately forecasting passenger satisfaction.

The key objectives of this study are specified below:

- To get the annotated dataset from secondary sources that includes the user-generated content (online reviews) and its associated passenger satisfaction category.
- Using pre-processing techniques on internet reviews, such as padding, producing word embeddings, and eliminating punctuation.
- Encrypting the labels that go along with the reviews and storing them separately.
- To set up the validation and training data loaders.
- Setting up the suitable model parameters such as optimiser, batch size, epochs, scheduler, learning rate, performance metrics.
- Using the provided datasets from online reviews, develop, train, and assess the BERT model for predicting passenger happiness.
- To evaluate the accuracy, recall, precision, and F1 Score values of the built-in classifier using the Confusion Matrix and Classification Report.
- To verify the model by producing the passenger satisfaction category by passing the textual reviews on unseen data.

1.4 Applications of Airline Satisfaction Predictor

Many users and applications are possible for the BERT-based airline passenger satisfaction model, including:

Airlines: Airlines can use the model to examine consumer comments and evaluations, pinpoint the elements that influence passenger satisfaction, and customise their offerings to suit the demands of individual clients. The approach can also assist airlines in increasing their competitiveness and financial performance by assessing the efficacy of their customer service procedures and rules.

Aviation industry analysts: Aviation industry analysts can use the model to evaluate the effectiveness of various airlines based on ratings of customer satisfaction and to pinpoint the characteristics that set high-performing airlines apart from those that perform poorly. This can help with investment choices and support the aviation industry's overall sustainability and expansion.

Passengers: The model benefits passengers by ensuring that the airlines take into account their comments and suggestions regarding airline services and experiences. This may help to raise customer happiness while also raising the level of service quality and overall passenger experience.

Aviation regulators and policymakers: Regulators and policymakers in the aviation industry can use the model to assess the level of airline services and pinpoint areas that may require regulatory action to safeguard the rights and interests of passengers. The model can help with the creation of laws and policies that support ethical business practises, customer safety, and long-term growth in the aviation sector.

Travel and Tourism Organizations: Travel and tourism organisations can use this technology to better understand the requirements and expectations of their customers, which enables them to make data-driven decisions about things like which airlines to partner with or promote.

Academic Researchers: This study can be interesting for other academic scholars because it examines the application of sentiment analysis in the aviation industry.

Ultimately, the BERT-based airline customer satisfaction model can help a variety of aviation industry stakeholders and can enhance the overall passenger experience as well as the competitiveness and sustainability of the sector.

1.5 Challenges in Airline Satisfaction Predictor

There are various difficulties in creating a BERT predictor for airline passenger satisfaction, including:

Data gathering and pre-processing: Because the data may be overwhelming, chaotic, and loud, gathering and preparing passenger reviews and comments can be difficult. To make sure the data is correct and useful, cleaning and pre-processing can be time-consuming and require subject knowledge.

Domain-specific knowledge: Knowledge of the aviation business as a whole, including the elements that influence passenger satisfaction, such as flight delays, baggage handling, in-flight services, and customer assistance, is necessary to develop a BERT model that accurately forecasts airline passenger satisfaction. It can be difficult to incorporate this knowledge into the model, thus specialists in the field must work together.

Training and fine-tuning the BERT model can be time-consuming and expensive in terms of computing, especially when working with huge datasets. In order to increase the model's precision in forecasting customer pleasure, it may need to be adjusted for certain airline domains and languages.

Fairness and bias: The BERT model may not be fair if it was not trained on a variety of data or if the training data used were not representative of the population. Furthermore, if the model's projections unfairly disadvantage some passenger groups, such as those with impairments, low-income customers, or passengers from particular racial or ethnic groups, they may not be accurate.

Interpretability: Predictions made by the BERT model may be challenging to understand, particularly if the model is complicated and was developed using a significant quantity of data. Making decisions based on the model's predictions can be difficult as a result, making it difficult to pinpoint the variables that influence passenger happiness.

By overcoming these obstacles, the BERT-based airline passenger satisfaction predictor can offer insightful information about customer satisfaction, help to shape industry rules and practises, and improve the overall passenger experience.

1.6 Overview of This Report

The report comprises of eleven chapters. Chapter 1 presents the Introduction to the project. The theoretical framework of BERT model along with recent work has been discussed in Chapter 2. Chapter 3 presents the research methodology along with the working of BERT model. Chapter 4 discusses the Requirements in order to make this work. Chapter 5 discusses the design of the work. Chapter 6 presents the Implementation part of the work. Chapter 7 discusses the Testing and Results. Chapter 8 presents the Project Management section which includes project schedule, risk management, quality management. Chapter 9 presents the Critical appraisal of this work. Chapter 10 presents the Conclusion. Student Reflections are discussed in Chapter 11.

2 Literature Review

2.1 Theoretical Framework

2.1.1 BERT Introduction

BERT stands for Bidirectional Encoder Representations from Transformers. It is a deep learning model that has already been trained and is utilised for natural language processing tasks. BERT is a powerful tool for a range of natural language processing tasks, including language understanding, text classification, and question-answering. It is built on the transformer architecture and pre-trains the model using a substantial quantity of text data.

There are two primary categories of BERT- BERT-base and BERT-large. The smallest of the two models, **BERT-base** has 12 layers and 110 million parameters. This approach is frequently used for tasks like text categorization or sentiment analysis that call for less intricate language understanding. **BERT-large** is the bigger of the two models, with 24 layers and 340 million parameters. This paradigm is frequently employed for harder natural language processing tasks like language inference, summarization, and answering questions.

There are two another variety of BERT: cased and uncased. In **cased BERT**, letters are processed differently depending on their case, therefore uppercase and lowercase letters are treated separately. When working with languages like German or French, where a word's case can alter its meaning, this is helpful. In **uncased BERT**, every letter in the text is changed to lowercase before processing, thus uppercase and lowercase letters are treated equally. When working with languages like English, where a word's case typically has little bearing on its meaning, this is helpful. Both the cased and uncased BERT models offer benefits and drawbacks. Uncased BERT may be more appropriate for languages where case is less significant, whereas cased BERT may be more suited for languages where case is more significant. Uncased BERT is often more widely utilised since it is more adaptable and suitable for a wider range of natural language processing jobs.

In this work, text classifier has been developed using the **BERT-base-uncased** model.

2.1.2 Advantages of BERT model

Pre-trained on massive datasets: BERT is very effective for a range of natural language processing applications because it has been pre-trained on a large corpus of text data.

Bi-directional: BERT can capture contextual relationships between words better since it is bi-directional, which means it considers both the words that come before and after it in a phrase.

Fine-tuning: BERT is highly adaptive and versatile, and it can be fine-tuned to handle a range of natural language processing jobs.

Outperforms traditional NLP models: BERT has been demonstrated to outperform conventional natural language processing models on a number of tasks, including sentiment analysis, text categorization, and question-answering.

Open-source: BERT is an open-source paradigm, allowing anybody to use and alter it for their own applications involving natural language processing.

BERT's pre-training on enormous datasets, bi-directional design, and capacity to be customised for particular use cases make it a very effective and adaptable natural language processing model that can be utilised for a range of applications.

2.1.3 BERT architecture

The transformer architecture, a style of neural network architecture that is particularly well-suited to natural language processing applications, provides the foundation for the architecture of BERT. There are two primary parts of the BERT model:

1. **Transformer Encoder**: The transformer encoder used by BERT is a multi-layered encoder made up of a stack of identical transformer blocks. Every transformer block has two lower layers:

- Self-Attention Layer: This layer uses attention processes to compute a weighted sum of the input sequence, creating a context vector for each token in the sequence. It takes a sequence of input embeddings as input.
- Feedforward Layer: This layer takes the output of the self-attention layer and applies a straightforward, position-wise fully linked feedforward network.

2. **Pre-training and Fine-tuning**: BERT is trained beforehand on a sizable corpus of text data, and the weights from that training are subsequently adjusted for particular tasks involving natural language processing. BERT is trained on two unsupervised activities during the pre-training phase: Modeling Masked Language and Predicting the Next Sentence. Taking the pre-trained weights and adjusting them for certain natural language processing tasks, such as text categorization, question-answering, and language inference, is the fine-tuning procedure.

BERT's overall design is based on the transformer architecture, which is particularly effective for jobs involving natural language processing. The model comprises of a multi-layered transformer encoder that has been specially adjusted for various natural language processing tasks after being pre-trained on a huge corpus of text data.

2.2 Traditional Methods of Text Classification

Traditional text categorization techniques include a number of machine learning and recurrent neural network-based techniques. Below is a quick breakdown of each strategy:

1. Methods based on machine learning: These techniques often start with feature extraction and selection, then train a machine learning algorithm to categorise text. Techniques for feature extraction that are often used include term frequency-inverse document frequency, n-grams, and bag-of-words (TF-IDF). Support vector machines, decision trees, and Naive Bayes are common machine learning techniques (SVMs).

2. Recurrent neural networks (RNNs): These are a type of neural network that are used in these techniques to classify text. For text classification tasks, RNNs' capacity to process sequential data and preserve a memory of prior inputs is helpful. Long short-term memory (LSTM) and gated recurrent unit (GRU) networks are typical RNN architectures used for text classification.

Challenges faced by these techniques- For text categorization problems, approaches based on recurrent neural networks and machine learning have both been successful. Yet, there are some challenges faced by these techniques which are discussed in this section.

1. Lack of data: To be trained successfully, text classification algorithms need a lot of high-quality labelled data. But gathering such information can be difficult and time-consuming, especially for highly specialised subjects or languages.

2. Feature engineering: Hand-crafted features like n-grams, bag-of-words, and TF-IDF are frequently needed by machine learning-based approaches for classifying text. The choice of the proper features might be difficult, time-consuming, and call for understanding of the relevant subject.

3. Long-term dependencies: Recurrent neural network-based approaches are made to handle sequential data, but they may have trouble capturing long-term dependencies in text data. This may reduce their usefulness in particular text classification tasks.
4. Interpretability: Text categorization algorithms based on deep learning can be very complex, making it challenging to comprehend how they create predictions. This may limit its applicability in some situations where interpretability is crucial.

5. Resources needed: Deep learning-based text classification techniques frequently need a lot of computing power, including GPUs and plenty of memory. This may make it difficult for them to be adopted in some circumstances.

2.3 Recent Work

Shah and Sharma (2020) conducted sentiment analysis on a dataset of 20,000 tweets. Depending on their emotion, the tweets were classified as good, negative, or neutral. By eliminating stop words, stemming, and extracting features using the bag-of-words method, the authors pre-processed the data. The preprocessed dataset was then used to train both **Naive Bayes Classifier and Support Vector Machine** models, and their performance was assessed using metrics including accuracy, precision, recall, and F1-score. The outcomes demonstrated that, in terms of each of these metrics, the Naive Bayes Classifier outperformed the Support Vector Machine. More specifically, the Support Vector Machine had an accuracy of 72.45% compared to the Naive Bayes Classifier's 76.25%.

The performance of various machine learning techniques for sentiment analysis of Twitter data was compared by the authors in Chakraborty, Chakraborty, & Banerjee, 2021. The effectiveness of the **Logistic Regression, Random Forest, Support Vector Machine, Naive Bayes Classifier, and Decision Tree Classifier** models was specifically assessed. By eliminating stop words, stemming, and extracting features using the bag-of-words method, the authors pre-processed the data. In terms of accuracy, precision, recall, and F1-score, the results demonstrated that Random Forest performed better than the other models. In particular, Random Forest's accuracy was 75.1%, while Support Vector Machine's accuracy was 72.5%, making it the second-best model.

The performance of various RNN models for text classification on the two distinct datasets Reuters-21578 and Amazon Reviews containing millions of reviews was compared by the authors in their study (Wang et al.,2022). They assessed the effectiveness of the **Simple RNN, LSTM, GRU, and BiLSTM** models specifically. The outcomes demonstrated that the BiLSTM model performed better than the competing models on both datasets in terms of accuracy, precision, recall, and F1-score. The accuracy of the BiLSTM was 97.48% on the Reuters-21578 dataset, whereas the accuracy of the Simple RNN was 94.89%. The BiLSTM had an accuracy of 96.72% on the Amazon Reviews dataset, whereas the Simple RNN had an accuracy of 94.21%.

The complicated syntax and morphology of the Turkish language make sentiment analysis of Turkish texts a difficult undertaking. Arslan & Elibol, 2021 specifically tested the effectiveness of a **Long Short-Term Memory (LSTM)** network using word embeddings as inputs on a dataset for Turkish sentiment analysis. The outcomes shown that the suggested method outperformed other widely used models like Support Vector Machine (SVM) and Naive Bayes (NB) classifiers to attain state-of-the-art performance on the dataset. The accuracy of the LSTM-based model was 82.6%, whereas that of the SVM and NB models was 77.8% and 76.6%, respectively.

Abualsaud et al.,2021 used online reviews to predict airline passenger happiness using RNNs. The researchers employed a Deep Recurrent Neural Network (DRNN) model containing **Long Short-Term Memory (LSTM)** cells to classify a dataset of more than 5,000 airline reviews from the TripAdvisor website. The outcomes demonstrated that the DRNN model outperformed other conventional machine learning models including Support Vector Machines (SVMs) and Naive Bayes (NB) classifiers in predicting passenger pleasure, with a high accuracy of 92.76%.

Liu et al., 2022 used **BERT** model to predict the text classification on 6000 reviews dataset obtained from TripAdvisor website. The findings demonstrated that the BERT model outperformed other conventional machine learning models including Support Vector Machines (SVMs) and Naive Bayes (NB) classifiers in predicting passenger pleasure, with a high accuracy of 93.4%.

Gupta et al.,2022 compared the performance of **BERT and LSTM** models for text classification on three different datasets namely AG News dataset, the IMDB Reviews dataset, and the Stanford Sentiment Treebank (SST) dataset. The authors observed that the BERT model achieved an accuracy of 93.5% on the AG News dataset, 94.3% on the IMDB Reviews dataset, and 89.7% on the SST dataset, while the LSTM model achieved accuracies of 92.2%, 91.8%, and 87.9% on the same datasets, respectively. The study also reported that the BERT model required significantly less training time and fewer training epochs than the LSTM model to achieve better performance.

2.4 Summary of Recent Work

After reviewing numerous works, it can be observed that large number of text classification tasks are performed using algorithms namely LSTM, Naïve Bayes, Support Vector Machine, and BERT model. From the above analysis, it is crystal clear that BERT model has surpassed all the remaining algorithms for text classification. However, the work done by previous authors are not comparable to each other since the dataset plays a significant role in determining the accuracy of the work. As all the authors have worked on different datasets, thus, the findings of work cannot be compared with each other.

However, it can also be seen that majority of the work have experimented with more than one technique for classifying texts. It is noticeable in those works that BERT model has given better results from LSTM which is a recurrent neural network (Gupta et al.,2022). Also, it has been observed that LSTM or other machine learning algorithms require a significant amount of time and resources for pre-processing and training which is no longer required in BERT model.

Thus, after reviewing several recent works, this work aims to analyse the performance of BERT model for predicting the airline passenger satisfaction using the user generated content.

3 Methodology

This section delves into the discussion of research methodology used for the fulfilment of this work and BERT methodology on which the entire experimental setup is based on.

3.1 Research Methodology

The research methodology used in developing this work is CRISP-DM which stands for the CRoss Industry Standard Process for Data Mining (CRISP-DM). The methodology can be applied via following the below specified steps. These steps can also be visualised from the figure shown below.

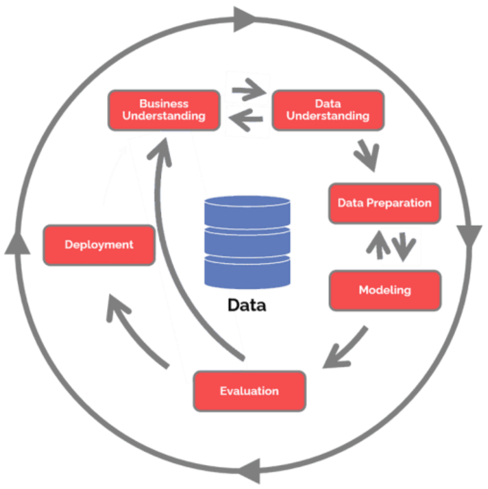


Figure 1 CRISP-DM methodology

1. **Business Understanding**: In this phase, all the factors are analysed that could have an impact on client satisfaction, which could leave a bad impression and lead to unhappiness or could inspire the consumer to keep returning. Realistic outcomes would follow from having such a grasp of all the factors that could influence the customer's opinion and how he might decide to book frequently. The resulting model might produce excellent results, but it might not be valid as the real variables are missing. So, it is crucial to concentrate on all scenarios that the passenger can encounter when travelling.

2. **Data Understanding**: At this point, the suitable data is initially searched from different available sources considering all the ethical and legal issues. Different features present in the dataset are then reviewed to gain the understanding of data.

3. **Data Preparation**: Once it has been obtained, the data is then being brought to consistent format by converting it into csv file. Some basic data cleaning steps are then applied on the user generated content (online reviews reflecting customer feedback). This involves removing punctuation, special characters. The data is the split into three subsets- Training, validation, and testing. Data preparation involves some more detailed steps as well such as pre-processing using BERT tokeniser, creating torch tensors, data loaders and many more. All of these steps are discussed in Implementation section of this work.

4. **Modeling**: The next step after preparing the data is to build and train the model. At this step, suitable model parameters such as number of epochs, batch size, performance metrics, optimiser, scheduler, learning rate, epsilon value, and many others are finalised. A pre-trained BERT model is then loaded which has been fine-tuned for the required classification task. During training,

classifier is passed with both inputs as well as output values. The model tries to learn generating the predictions for the input data using the forward and backward pass.

5. **Evaluation**: The implemented classifier is then evaluated with the validation data where predictions generated by the classifier is matched with the actual values to assess the performance of the implemented classifier. The performance of the classifier is analysed using the accuracy per class measure, confusion matrix, and classification report giving measures such as accuracy, recall, f1 score, and precision.

6. **Deployment**: Deployment is the last action which can be followed whilst embedding the application in real-world scenarios. This is the stage where all necessary tasks are completed and the model is verified to be well-optimized and producing accurate results. Also, if the outcomes meet the needs of the business, it is time to move the project into production. The deployment procedure requires both batch and online analysis to be able to function.

3.2 How BERT model works?

The steps involved in Text classification using BERT model are:

Tokenization: The input text must first be divided up into separate tokens. BERT employs WordPiece tokenization, which divides words into subwords and enables it to handle words that are not commonly used.

Creating Input IDs: Using a pre-defined vocabulary, the tokens are then mapped to the appropriate IDs. Moreover, each input sequence has a special token ([CLS]) prepended before it and another special token ([SEP]) appended after it to indicate the start and end of the input sequence, respectively.

Attention Masks: BERT also involves the generation of an attention mask for each input sequence. To distinguish between tokens that are padding and those that are a part of the input sequence, we employ the attention mask.

Embeddings: When the input sequence has been passed through several layers of transformer-based encoders, each token in the sequence is produced with an embedding. These embeddings use the words around each token to determine its context and meaning.

Pooling: For each token in the input sequence, the transformer encoder's ultimate output is a sequence of embeddings. A pooling operation is carried out on these embeddings to provide a fixed-size vector representation of the input sequence, which is then used as the output for text categorization. The pooled form of the input sequence used by BERT is the output of the [CLS] token.

Classification: To get the final class probabilities, the pooled representation of the input sequence is then processed through a fully connected layer and a softmax activation function.

A fixed-size vector representation of the input sequence that can be utilised for text classification is made possible by the embeddings in the BERT model, which are designed to capture the context and meaning of each token depending on the words around it. The padding tokens are distinguished from the input sequence using the attention masks, which enables the model to ignore them during training and inference.

4 Requirements

The following hardware and software specifications are necessary for creating a text classifier using the BERT model:

Software specifications:

Python: The BERT model is often implemented using the Python programming language. The BERT model can be implemented using any of the machine learning libraries provided by Python, including TensorFlow, PyTorch, and Keras.

Deep Learning Libraries: The most often used deep learning libraries for BERT model implementation are TensorFlow or PyTorch. They offer APIs that may be used to create, train, and assess the models. In this work, majority of the development is done using PyTorch library.

Transformers Library: The Transformers library is a well-liked library for putting BERT models into practise and optimising them. It offers models that have already been trained as well as APIs for modifying the models for particular tasks like text classification.

Programming Platform: Popular programming platforms for machine learning tasks include Jupyter Notebook and Google Colaboratory. On one hand, Google Colaboratory is a cloud-based interactive development environment that offers free access to GPUs for deep learning tasks. On the other hand, Jupyter Notebook is a web-based interactive development environment that allows running Python code in the browser.

Some Other Libraries: Popular Python data manipulation libraries Pandas and Numpy are used for preparing and analysing data. Some other libraries such as Matplotlib, Seaborn are also required for data analysis and generating visualisations. Sklearn library is required to analyse the performance of the implemented classifier using Confusion Matrix and Classification report. Tqdm library is required to generate the progress bar displayed when the classifier is being trained. Re and string libraries are required for doing basic data cleaning.

Hardware requirements:

GPU: BERT model training is computationally expensive and time-consuming, hence using a GPU is advised for speedier training. Little datasets can also be processed by a CPU; however the model training process could take longer.

Memory: Memory requirements for the BERT model are high; it is advised to have at least 8GB for smaller datasets and 16GB or more for larger datasets.

Storage: The dataset used for training and evaluation might be substantial, so it is advised to have enough storage space for the model checkpoints and data.

Operating System: Windows, Linux, and macOS are just a few of the common operating systems that are compatible with the software requirements.

In general, you need a computer with the hardware resources and the necessary software libraries installed in order to create a text classifier using the BERT model. If one don't have access to a powerful machine, one can also use a cloud-based development environment like Google Colaboratory. There are numerous advantages of using a cloud-based environment Google Colab as it offers GPU access, large storage, RAM. Also, there is no need to externally install the software and libraries. Thus, in this work, Google Colaboratory has been used for experimental purposes.

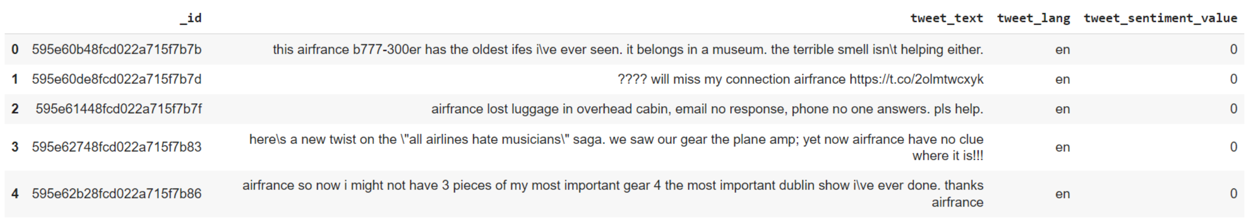
4.1 Dataset Description

A potent natural language processing (NLP) model called BERT (Bidirectional Encoder Representations from Transformers) can be applied to text categorization challenges. However, BERT needs data to be trained on, much like any machine learning model. When utilising BERT for text classification, the goal of a dataset is to give the model instances of input text and their related labels or categories. The model will be able to perform better on new, untested data the more varied and representative the dataset is of real-world circumstances. A BERT model cannot be trained for text classification without a dataset since it has no instances from which to learn. The model may learn patterns and correlations between words and phrases in the input text and their associated labels with the help of the dataset, which serves as the model's training ground. Having a dataset is essential for creating a text classifier using BERT since it enables the model to learn from instances and gradually increase its accuracy.

Thus, a publicly available dataset has been downloaded from OpenML repository which comprises of large number of datasets suitable for research and study purposes. The dataset downloaded from the OpenML repository is not in consistent manner. It was a txt file. In order to train classifiers, it is good to use csv files so that tabular data can be analysed easily. The csv file is then prepared from the downloaded file. The dataset comprises of four columns namely- id, tweet_text, tweet_lang, tweet_sentiment_value.

- Id column represents the unique identifier for each tweet or review present in the dataset.
- Tweet_text column contains the review given by the user reflecting the customer experience or feedback for the airline.
- Tweet_lang shows the language in which feedback has been given by the user. It is visible that all the users have given the feedback in English language.
- Tweet_sentiment_value reflects the value of sentiment of review or the passenger satisfaction level. It comprises the value in the form of integers 0,1, and 2 where 0 represents the negative experience, 1 represents the neutral experience, and 2 represents the positive experience.

After preparing a csv file, the file has been imported to the google colab environment where all the experiments are performed. After uploading the csv file, it is being read using the python functions and a pandas dataframe has been created using the csv file to fulfil experimental needs. The first five rows of the imported data stored in dataframe are shown below in the figure. The dataset comprises of 1097 tweets. It can also be seen that there are so much unnecessary punctuation marks, special characters present in the review. Thus, some basic data cleaning is required prior to pass these text values to BERT tokenizer for pre-processing.



	_id	tweet_text	tweet_lang	tweet_sentiment_value
0	595e60b48fcd022a715f7b7b	this airfrance b777-300er has the oldest ifes i\ve ever seen. it belongs in a museum. the terrible smell isn\t helping either.	en	0
1	595e60de8fcd022a715f7b7d	???? will miss my connection airfrance https://t.co/2olmtwcxyk	en	0
2	595e61448fcd022a715f7b7f	airfrance lost luggage in overhead cabin, email no response, phone no one answers. pls help.	en	0
3	595e62748fcd022a715f7b83	here\s a new twist on the \"all airlines hate musicians\" saga. we saw our gear the plane amp; yet now airfrance have no clue where it is!!!	en	0
4	595e62b28fcd022a715f7b86	airfrance so now i might not have 3 pieces of my most important gear 4 the most important dublin show i\ve ever done. thanks airfrance	en	0

Figure 2 Preview of imported Data

There are two major columns of interest in the fetched data namely- tweet_text, and tweet_sentiment_value as these contain the user generated content and its associated label reflecting customer satisfaction. The target class is tweet_sentiment_value in this work.

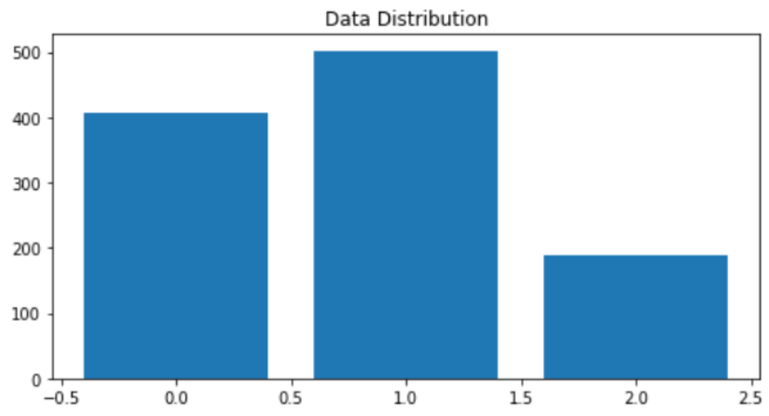


Figure 3 Countplot showing the Target Class Distribution

The class distribution of target class- tweet_sentiment_value is shown above in the figure. There are 502 reviews belonging to negative class (0), 406 reviews belonging to neutral class (1), and 189 reviews belonging to positive class (2).

This leads to the conclusion that dataset is quite imbalanced in the nature as there are very few samples belonging to positive customer experience in comparison to negative or neutral. However, this imbalance is handled during experimental purposes discussed in Implementation section.

5 Design

This section discusses the high-level overview of the steps taken to complete this work. The image below shows all the vital steps taken in the form of block diagram.

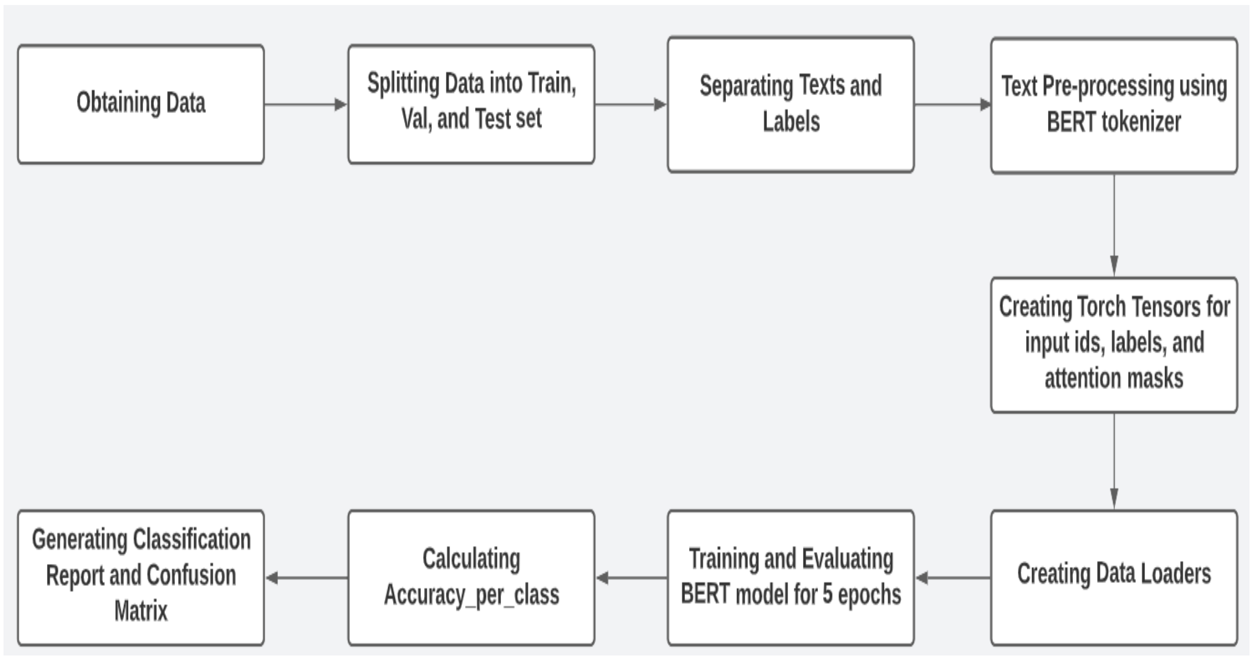

Obtaining Data
Splitting Data into Train, Val, and Test set
Separating Texts and Labels
Text Pre-processing using BERT tokenizer
Creating Torch Tensors for input ids, labels, and attention masks
Creating Data Loaders
Training and Evaluating BERT model for 5 epochs
Calculating Accuracy_per_class
Generating Classification Report and Confusion Matrix

Figure 4 Block Diagram illustrating Design

The purpose of this work is to develop a model that can facilitate aviation industries in improving their client services after analysing customer reviews and feedbacks. BERT model has been used to implement the classifier. In order to utilise BERT model for generating predictions, the BERT model needs to be trained with suitable amount of textual data where it identifies the relationship between inputs and outputs.

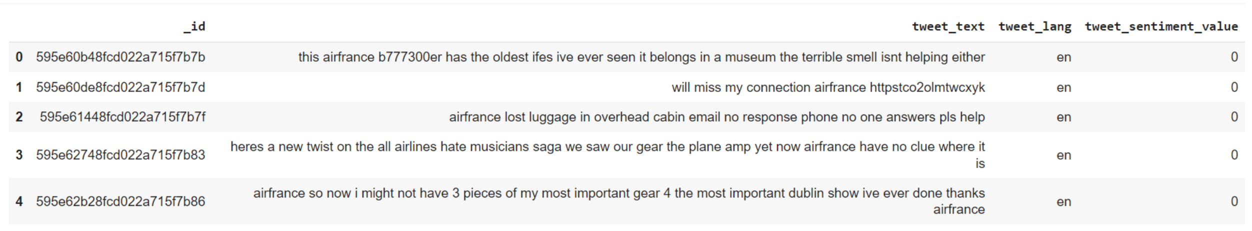
In order to train the BERT classifier, the data has been fextched from the OpenML repository. The dataset comprises of 1097 tweets belonging to three different categories- ‘Negative’, ‘Neutral’, or ‘Positive’. Thus, the model is also aimed to assign any of the possible three categories reflecting the passenger satisfaction level.

Different steps have been followed to implement a text classifier which accepts inputs as the user generated online customer review reflecting user experience and generates the passenger satisfaction level as output. The major involved in developing such classifier are Data Cleaning using BERT Tokenizer (Generating input ids, attention masks, and pad sequences) and generating predictions for the pre-processed user review. These steps are further discussed more broadly in the following Implementation section and also shown in block diagram above.

6 Implementation

Basic Data Cleaning Steps- After reading the dataset, basic data cleaning steps are performed. These steps include checking for null values, displaying the counts of review for each category- ‘positive’, ‘negative’, and ‘neutral’. There are no null values identified in the dataset.

The next step is to remove several tags such as ‘<, ^, ?, +’ , punctuation, and some special characters. The image below shows the top 5 rows of the dataframe obtained after basic data cleaning steps.



	_id	tweet_text	tweet_lang	tweet_sentiment_value
0	595e60b48fcd022a715f7b7b	this airfrance b777300er has the oldest ifes ive ever seen it belongs in a museum the terrible smell isnt helping either	en	0
1	595e60de8fcd022a715f7b7d	will miss my connection airfrance httpstco2olmtwcxyk	en	0
2	595e61448fcd022a715f7b7f	airfrance lost luggage in overhead cabin email no response phone no one answers pls help	en	0
3	595e62748fcd022a715f7b83	heres a new twist on the all airlines hate musicians saga we saw our gear the plane amp yet now airfrance have no clue where it is	en	0
4	595e62b28fcd022a715f7b86	airfrance so now i might not have 3 pieces of my most important gear 4 the most important dublin show ive ever done thanks airfrance	en	0

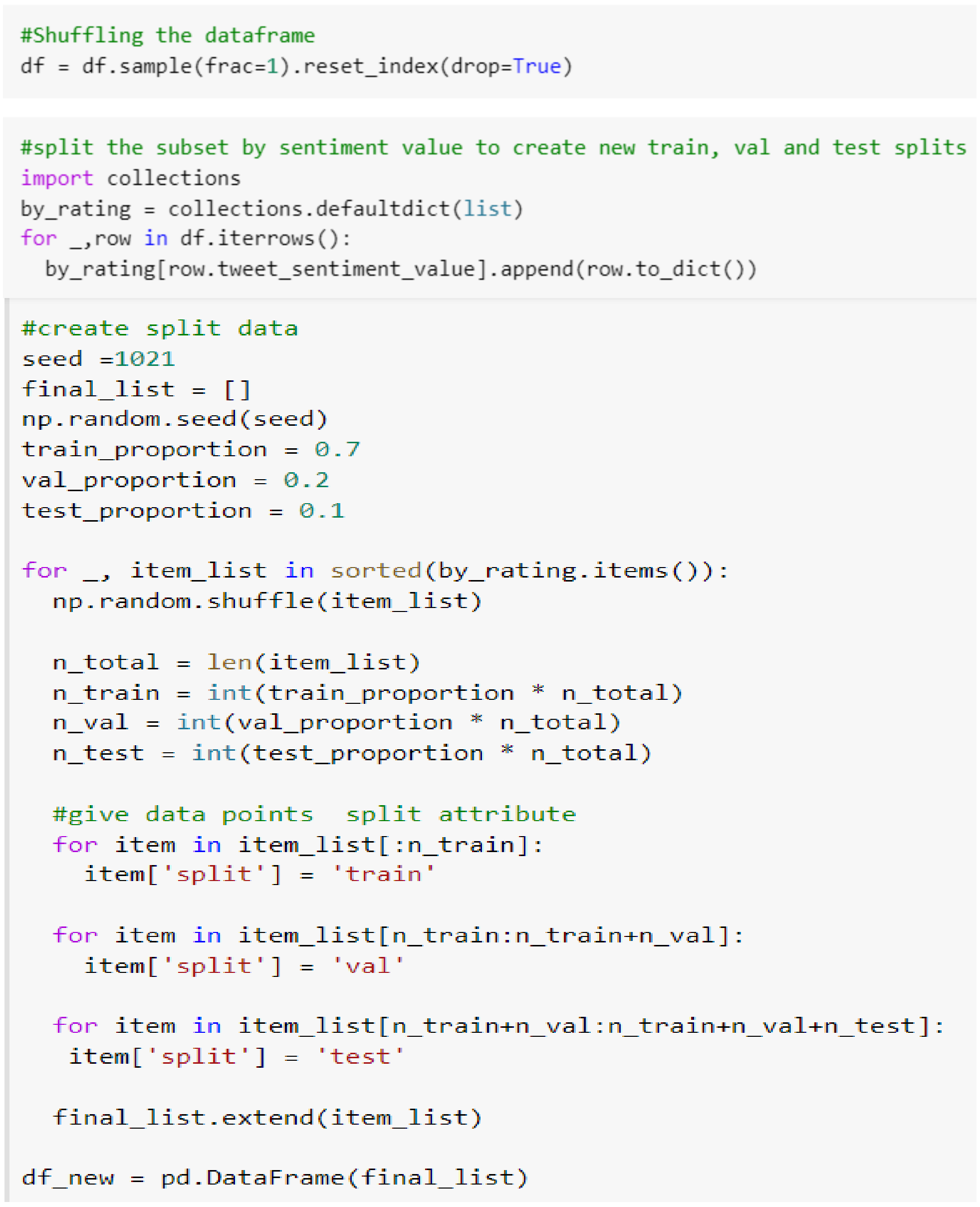
Figure 5 Dataframe after Data Cleaning

Preparing Training and Testing Data- The next step after basic data cleaning is to split the original data into training, validation, and testing data. As discussed above, the number of reviews present for each category of review are not comparable, thus the dataset is imbalanced. Therefore, these splits need to be done very carefully to eliminate the biased decisions from the model.

The dataset is split into train, validation, and test sets based on the proportion specified for each set to handle the dataset imbalance. The ‘by_rating’ dictionary is used to help in balancing the data to some extent by ensuring that each split has approximately the same proportion of samples from each class. It is not advisable to use techniques such as Oversampling and Undersampling for handling class imbalance of textual data as the performance of the model can be impacted and important information can also be lost. Textual data comprises of rich and complex structures which can be lost while making use of techniques such as Oversampling or Undersampling.

Each class in text classification could contain a varied amount of samples, but that does not indicate the model will necessarily do poorly on the minority class. To handle class imbalance in text classification tasks, alternative approaches like class weighting, creating synthetic data, or utilising a different metric like F1 score might be used. However, in some particular circumstances, such as when the dataset is too huge or when the minority class has a very small number of samples, oversampling or undersampling may be required. When dealing with class imbalance in text data, it is advised to utilise strategies like SMOTE (Synthetic Minority Over-sampling Technique) for oversampling or Tomek linkages for undersampling.

The code snippet below shows how the splits are generated from the original dataframe.



```
#Shuffling the dataframe
df = df.sample(frac=1).reset_index(drop=True)
```

```
#split the subset by sentiment value to create new train, val and test splits
import collections
by_rating = collections.defaultdict(list)
for _,row in df.iterrows():
  by_rating[row.tweet_sentiment_value].append(row.to_dict())
```

```
#create split data
seed =1021
final_list = []
np.random.seed(seed)
train_proportion = 0.7
val_proportion = 0.2
test_proportion = 0.1

for _, item_list in sorted(by_rating.items()):
  np.random.shuffle(item_list)

  n_total = len(item_list)
  n_train = int(train_proportion * n_total)
  n_val = int(val_proportion * n_total)
  n_test = int(test_proportion * n_total)

  #give data points  split attribute
  for item in item_list[:n_train]:
    item['split'] = 'train'

  for item in item_list[n_train:n_train+n_val]:
    item['split'] = 'val'

  for item in item_list[n_train+n_val:n_train+n_val+n_test]:
   item['split'] = 'test'

  final_list.extend(item_list)

df_new = pd.DataFrame(final_list)
```

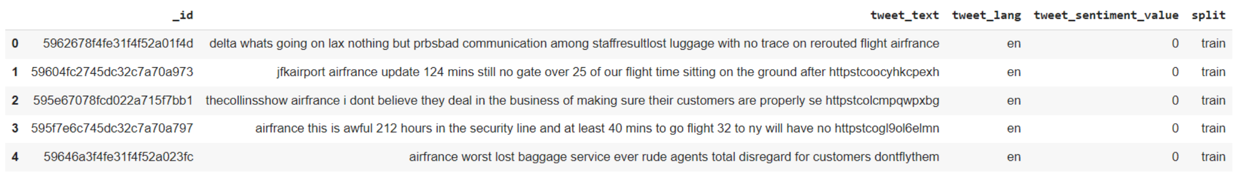
Figure 6 Code Snippet illustrating Dataset Splitting Process

Firstly, the original dataset has been shuffled randomly by calling the sample() method with frac=1, which returns all the rows in the shuffled order. Then, the dataset is split into subsets based on the values of the target variable (tweet_sentiment_value). This is done using a *defaultdict* from the *collections* module, which creates a dictionary where each key corresponds to a unique value of the target variable and the values are lists of rows with that target value.

After this, random seed is set to 1021, and three subsets namely training, validation and testing sets are formed using the specified proportions of train_proportion, val_proportion and test_proportion. The n_total variable is set to the total number of rows in the subset, and the n_train, n_val, and n_test variables are set to the number of rows that should be in each set based on the proportions.

Next, the code adds a new column called split to each row in the subset to indicate which set it belongs to. Rows in the training set are marked with train, rows in the validation set are marked with val, and rows in the testing set are marked with test.

Finally, all the subsets are concatenated into a new dataframe called df_new. The resulting dataframe has the same number of rows as the original dataframe, but with a new split column indicating which set each row belongs to. The resulting dataframe is shown in the image below.



	_id	tweet_text	tweet_lang	tweet_sentiment_value	split
0	5962678f4fe31f4f52a01f4d	delta whats going on lax nothing but prbsbad communication among staffresultlost luggage with no trace on rerouted flight airfrance	en	0	train
1	59604fc2745dc32c7a70a973	jfkairport airfrance update 124 mins still no gate over 25 of our flight time sitting on the ground after httpstcoocyhkcpexh	en	0	train
2	595e67078fcd022a715f7bb1	thecollinsshow airfrance i dont believe they deal in the business of making sure their customers are properly se httpstcolcmpqwpxbg	en	0	train
3	595f7e6c745dc32c7a70a797	airfrance this is awful 212 hours in the security line and at least 40 mins to go flight 32 to ny will have no httpstcogl9ol6elmn	en	0	train
4	59646a3f4fe31f4f52a023fc	airfrance worst lost baggage service ever rude agents total disregard for customers dontflythem	en	0	train

Figure 7 Dataframe comprising of Split column

It can be seen from above figure showing top 5 rows of the resulting dataframe that split column has been added which signifies the set in which review is appended. The distribution of these reviews are also analysed using the bar chart shown below.

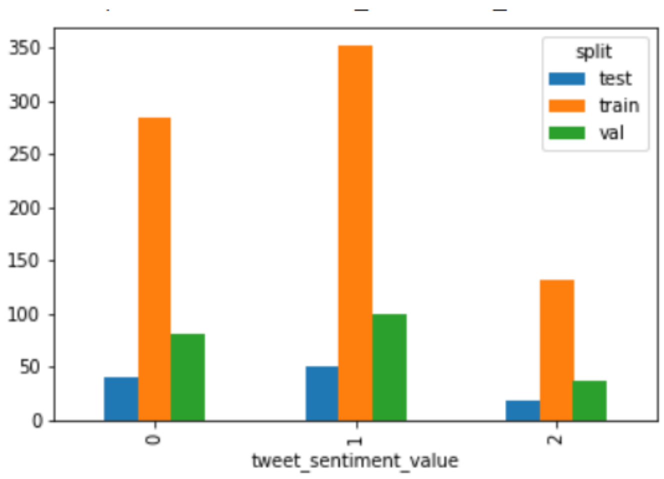


Figure 8 Bar Chart showing the Subset distribution

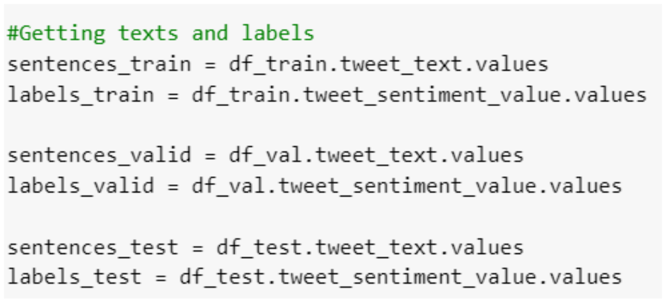
For sentiment value 0 (Negative), there are 284 instances in the training set, 81 instances in the validation set, and 40 instances in the test set. For sentiment value 1 (Neutral), there are 351 instances in the training set, 100 instances in the validation set, and 50 instances in the test set. And for sentiment value 2 (Positive), there are 132 instances in the training set, 37 instances in the validation set, and 18 instances in the test set. The information can also be visualised from the table shown below.

Type of Review	Subset Type	Number of Reviews
Negative (0)	Train	284
	Validation	81
	Test	40
Neutral (1)	Train	351
	Validation	100
	Test	50
Positive (2)	Train	132
	Validation	37
	Test	18

Table 1 Class Distribution in Training, Validation, and Testing

There are total 767 samples present in the training set, 218 samples present in validation set, and 108 samples present in testing set. Different datasets are then formed for each subset namely- df_train, df_val, and df_test.

Separating Texts and Labels- The next step after forming splits from the dataset is to separate the textual data and their associated labels so that these can be used for training, evaluating, and testing the classifier. The value from the column tweet_text has been fetched from all the subsets and stored in variables sentences_train, sentences_valid, and sentences_test. Similarly, the values from the column tweet_sentiment_value are fetched from the dataframes and stored in variables labels_train, labels_valid, and labels_test. The below figure shows the code block used to separate and store the text and labels.

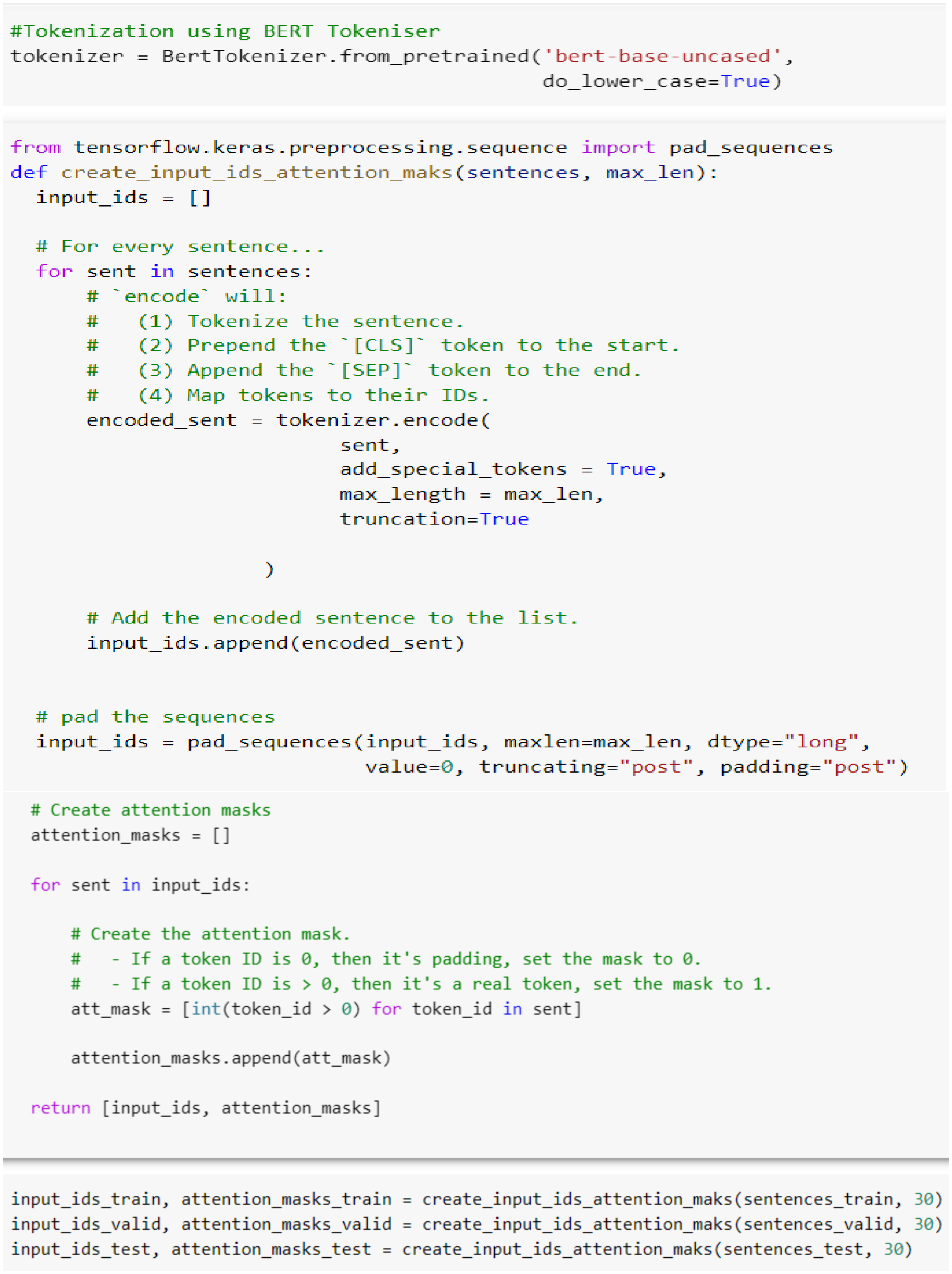


```
#Getting texts and labels
sentences_train = df_train.tweet_text.values
labels_train = df_train.tweet_sentiment_value.values

sentences_valid = df_val.tweet_text.values
labels_valid = df_val.tweet_sentiment_value.values

sentences_test = df_test.tweet_text.values
labels_test = df_test.tweet_sentiment_value.values
```

Figure 9 Code Snippet illustrating separately storing text and labels

Text pre-processing using BERT tokenizer- The next step after separating the texts and labels is to pre-process these text values so that they can be used for model training purposes. During pre-processing, input IDs and attention masks are generated using the BERT tokenizer. Each input text is represented using an integer assigned to input ID. Real tokens and padding tokens are distinguished using the attention masks. This section presents the step-wise pre-processing of textual data.

- Initially import BERT tokenizer (‘bert-base-uncased’) from the Hugging Face Transformers library.
- For padding the sequences with 0, import the pad_sequences function from the Keras library.
- A function named ‘create_input_ids_attention_masks’ has been created that accepts the list of sentences for pre-processing along with the maximum length of sentences. This function returns the input ids and attention masks of the passed sentences.
    - Initialize an empty list input_ids to store the input IDs for each sentence.
    - For each sentence in the input list, tokenize the sentence using the BERT tokenizer, add special tokens ([CLS] and [SEP]) to mark the beginning and end of the sentence, and truncate or pad the sentence to the specified maximum length.
    - Append the encoded sentence to the input_ids list.
    - Pad the sequences with 0s so that all sequences have the same length.
    - Initialize an empty list attention_masks to store the attention masks for each sentence.
    - For each sequence in the input_ids list, create an attention mask that is 1 for real tokens and 0 for padding tokens.
    - Append the attention mask to the attention_masks list.
    - Return the input IDs and attention masks as a list.
- Call the create_input_ids_attention_maks function on the training, validation, and test datasets to get the input IDs and attention masks for each dataset.



```
#Tokenization using BERT Tokeniser
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased',
                                          do_lower_case=True)
```

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
def create_input_ids_attention_maks(sentences, max_len):
  input_ids = []

  # For every sentence...
  for sent in sentences:
      # `encode` will:
      #   (1) Tokenize the sentence.
      #   (2) Prepend the `[CLS]` token to the start.
      #   (3) Append the `[SEP]` token to the end.
      #   (4) Map tokens to their IDs.
      encoded_sent = tokenizer.encode(
                          sent,
                          add_special_tokens = True,
                          max_length = max_len,
                          truncation=True

                    )

      # Add the encoded sentence to the list.
      input_ids.append(encoded_sent)


  # pad the sequences
  input_ids = pad_sequences(input_ids, maxlen=max_len, dtype="long",
                            value=0, truncating="post", padding="post")
  # Create attention masks
  attention_masks = []

  for sent in input_ids:

      # Create the attention mask.
      #   - If a token ID is 0, then it's padding, set the mask to 0.
      #   - If a token ID is > 0, then it's a real token, set the mask to 1.
      att_mask = [int(token_id > 0) for token_id in sent]

      attention_masks.append(att_mask)

  return [input_ids, attention_masks]
```

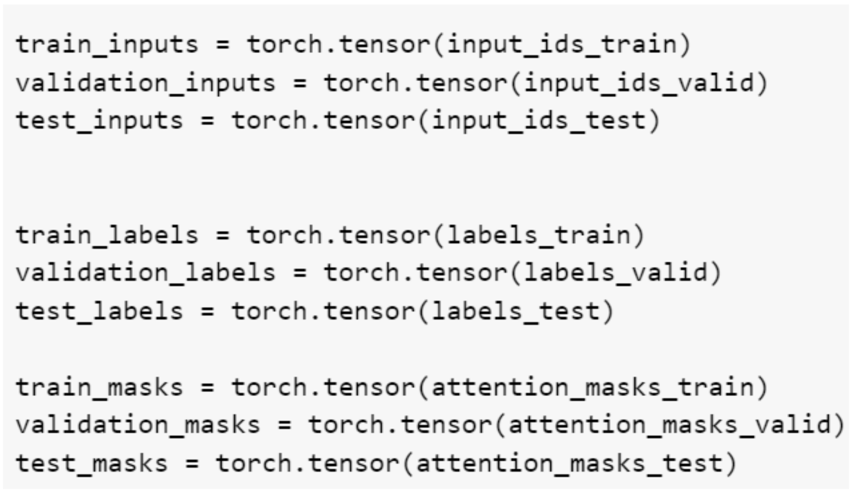
```
input_ids_train, attention_masks_train = create_input_ids_attention_maks(sentences_train, 30)
input_ids_valid, attention_masks_valid = create_input_ids_attention_maks(sentences_valid, 30)
input_ids_test, attention_masks_test = create_input_ids_attention_maks(sentences_test, 30)
```

Figure 10 Code Snippet showing the Text Pre-processing using BERT Tokenizer

Creating Torch Tensors- The next step after pre-processing text is to create torch tensors. PyTorch tensors are created for input data (train_inputs, validation_inputs, test_inputs), label data (train_labels, validation_labels, test_labels), and attention masks (train_masks, validation_masks, test_masks).

The input data (input_ids_train, input_ids_valid, input_ids_test) represents text data that has been pre-processed and tokenized into numerical representations in previous step. The label data (labels_train, labels_valid, labels_test) represents the target labels for this task which are 0 for negative, 1 for neutral, and 2 for positive.

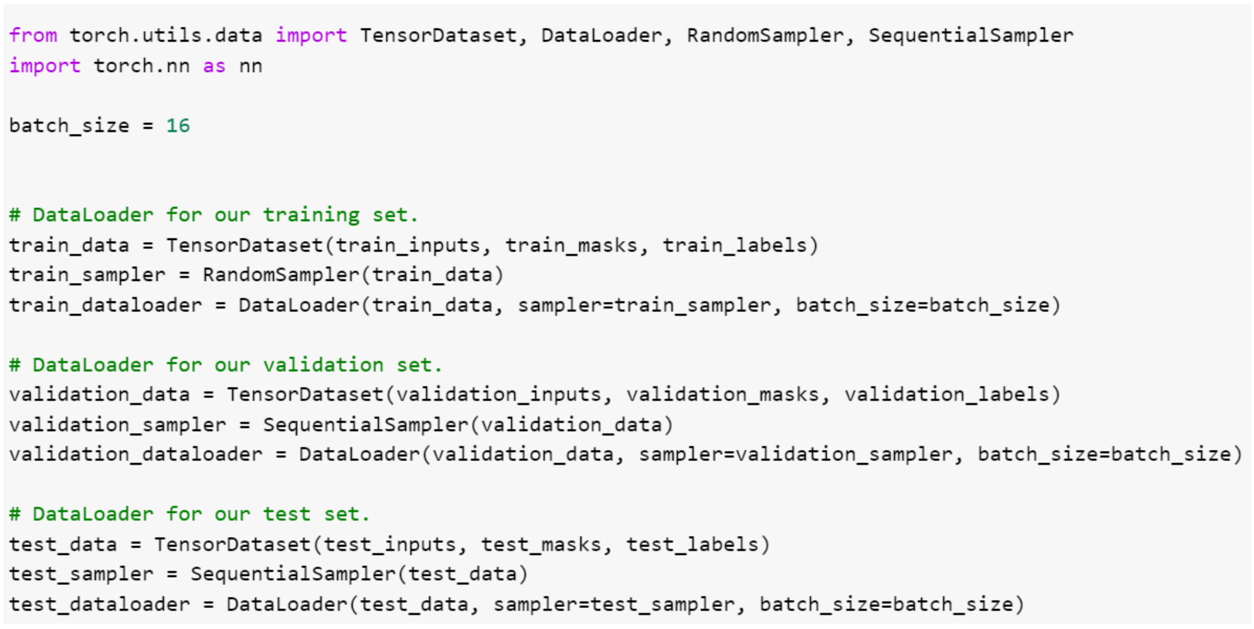
The attention masks (attention_masks_train, attention_masks_valid, attention_masks_test) are used in the Transformer model to indicate which tokens in the input sequence should be attended to and which should be ignored.



```
train_inputs = torch.tensor(input_ids_train)
validation_inputs = torch.tensor(input_ids_valid)
test_inputs = torch.tensor(input_ids_test)


train_labels = torch.tensor(labels_train)
validation_labels = torch.tensor(labels_valid)
test_labels = torch.tensor(labels_test)

train_masks = torch.tensor(attention_masks_train)
validation_masks = torch.tensor(attention_masks_valid)
test_masks = torch.tensor(attention_masks_test)
```

Figure 11 Code Snippet showing process of creating Torch tensors

Creating Data Loaders- The next step after creating torch tensors for input ids, attention masks, and labels is to create data loaders using these torch tensors. The image below represents the code snippet used to form the data loaders.



```
from torch.utils.data import TensorDataset, DataLoader, RandomSampler, SequentialSampler
import torch.nn as nn

batch_size = 16


# DataLoader for our training set.
train_data = TensorDataset(train_inputs, train_masks, train_labels)
train_sampler = RandomSampler(train_data)
train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=batch_size)

# DataLoader for our validation set.
validation_data = TensorDataset(validation_inputs, validation_masks, validation_labels)
validation_sampler = SequentialSampler(validation_data)
validation_dataloader = DataLoader(validation_data, sampler=validation_sampler, batch_size=batch_size)

# DataLoader for our test set.
test_data = TensorDataset(test_inputs, test_masks, test_labels)
test_sampler = SequentialSampler(test_data)
test_dataloader = DataLoader(test_data, sampler=test_sampler, batch_size=batch_size)
```

Figure 12 Code Snippet showing process of creating Data Loaders

Data loaders have been created for the training, validation, and test sets using the PyTorch *DataLoader* class. The purpose of using Data loader is to load data in batches during training and evaluating the BERT model.

Initially, necessary classes from PyTorch such as TensorDataset, DataLoader, RandomSampler, and SequentialSampler have been imported. The following step is to set the batch size which represents the number of samples model processes at a time during each iteration of training or evaluation. The batch size has been set to 16. The value of the batch size may vary considering the amount of computational resources available. It can be 32, 64, or any large value. Since, there is limited availability of GPU and RAM, the batch size is set to 16 in this work.

For the training set, a TensorDataset object has been created which takes in the training inputs, attention masks, and labels as arguments. A RandomSampler object is then created which randomly samples data from the training set to create batches during training. Finally, a DataLoader object is created using the TensorDataset and RandomSampler objects, setting the batch size to 16.

For the validation set, a similar process has been followed, but it creates a SequentialSampler object instead of a RandomSampler object. A SequentialSampler samples the data in order, which is useful for evaluation purposes as it ensures that all data is evaluated exactly once.

For the test set, a TensorDataset object is created using the test inputs, attention masks, and labels, and uses a SequentialSampler to sample the data in order. Finally, a DataLoader object is created using the TensorDataset and SequentialSampler objects, again setting the batch size to 16.

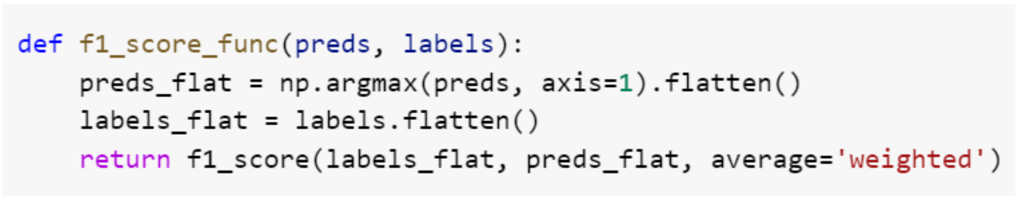
Setting Up model and parameters before Training- Initially it has been verified whether a CUDA-capable GPU is available and assigning the device accordingly for model building. If CUDA is not available, model is trained on CPU which takes a significant amount of time and is not advisable.

The next step after assigning a device is to **load a pre-trained BERT model** from transformers library- BertForSequenceClassification.from_pretrained(). This model has a single linear classification layer on top and uses the 12-layer BERT model with an uncased vocabulary. The num_labels parameter is set to 3, indicating that the model will classify inputs into one of three categories. The model is then moved to the assigned device using model.to(device).

An **optimizer** is created using the *AdamW* algorithm and the model parameters are passed to it. The **learning rate** is set to 2e-5 and the epsilon value is set to 1e-8. The number of epochs is set to 5.

Afterwards, get_linear_schedule_with_warmup() function has been used to create a **linear scheduler**. This scheduler adjusts the learning rate of the optimizer over time, starting with a warmup period where the learning rate gradually increases to the specified rate, and then decreases linearly over the remaining training steps. The num_warmup_steps parameter is set to 0, indicating that there will be no warmup period. The num_training_steps parameter is set to the total number of training steps (number of batches * number of epochs).

Setting Performance Metrics- Two performance metrics have been defined using functions- f1_score_func and accuracy_per_class. These functions are useful for determining the performance of classifier. These functions are defined in this section.



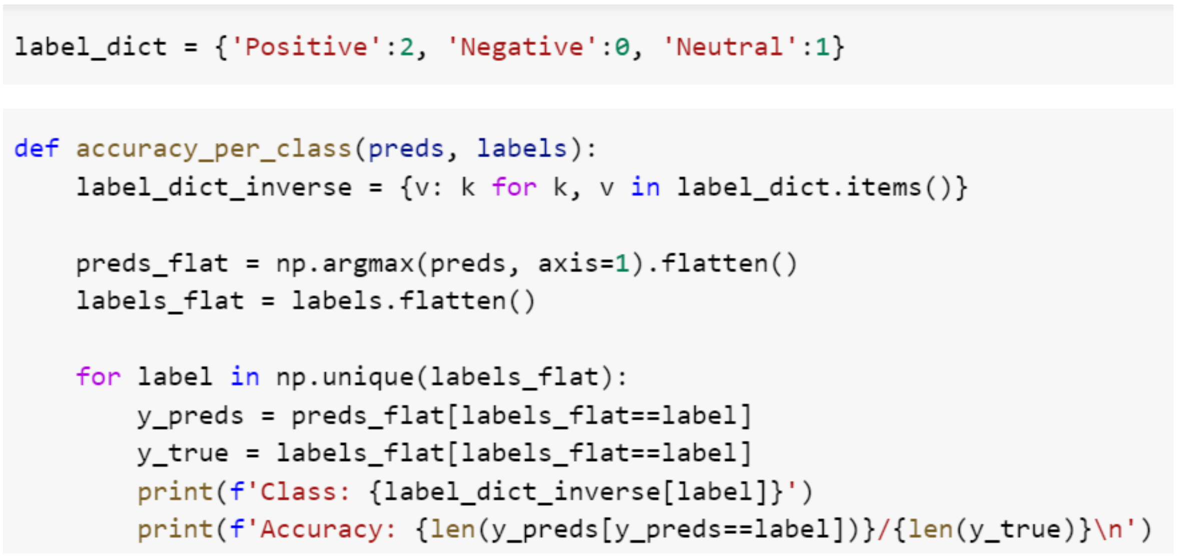
```
def f1_score_func(preds, labels):
    preds_flat = np.argmax(preds, axis=1).flatten()
    labels_flat = labels.flatten()
    return f1_score(labels_flat, preds_flat, average='weighted')
```

Figure 13 Function Definition - f1_score_func

The above figure shows a function definition for f1_score_func() that takes in two arguments: preds and labels. ‘preds’ are the predicted output values from the model, and ‘labels’ are the true target labels for the corresponding inputs.

A numpy.argmax() has been used to obtain the index of the highest predicted probability value for each input, resulting in a 1D array of predicted labels (preds_flat). The labels input is already a 1D array of true labels. The function then computes the weighted F1 score using sklearn.metrics.f1_score(), passing in the labels_flat and preds_flat arrays as well as the average parameter set to 'weighted'.

The weighted F1 score is a measure of the model's accuracy that takes into account class imbalance. It is the harmonic mean of precision and recall, weighted by the number of instances in each class. The weighted F1 score ranges from 0 to 1, with higher values indicating better performance.



```python
label_dict = {'Positive':2, 'Negative':0, 'Neutral':1}
```

```python
def accuracy_per_class(preds, labels):
    label_dict_inverse = {v: k for k, v in label_dict.items()}

    preds_flat = np.argmax(preds, axis=1).flatten()
    labels_flat = labels.flatten()

    for label in np.unique(labels_flat):
        y_preds = preds_flat[labels_flat==label]
        y_true = labels_flat[labels_flat==label]
        print(f'Class: {label_dict_inverse[label]}')
        print(f'Accuracy: {len(y_preds[y_preds==label])}/{len(y_true)}\n')
```

Figure 14 Function Definition- 'accuracy_per_class'

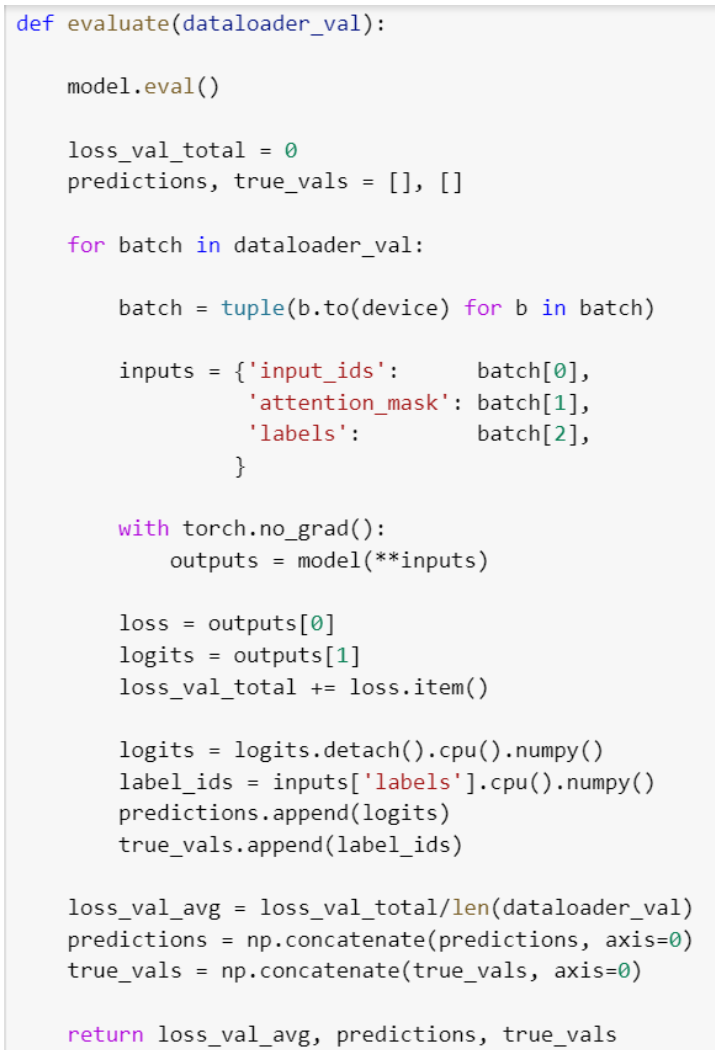
The above figure shows a function definition for accuracy_per_class() that takes in two arguments: preds and labels. A dictionary called label_dict has been used to map class names ('Positive', 'Negative', 'Neutral') to numerical labels (2, 0, 1). A new dictionary called label_dict_inverse is then created that maps numerical labels to class names.

A numpy.argmax() function has been used to obtain the index of the highest predicted probability value for each input, resulting in a 1D array of predicted labels (preds_flat). The labels input is already a 1D array of true labels. The unique values in labels_flat (i.e., the unique true labels) are then iterated and the accuracy for each class is calculated separately.

For each class, it selects the corresponding predicted labels (y_preds) and true labels (y_true) and computes the fraction of correct predictions. It then prints the class name along

The function prints the class name and accuracy for each class.

Defining Model Evaluation Function- After defining functions for performance metrics, the next step is to write a function definition for evaluating the model. The code snippet below shows the function defined for Model evaluation.



```
def evaluate(dataloader_val):

    model.eval()

    loss_val_total = 0
    predictions, true_vals = [], []

    for batch in dataloader_val:

        batch = tuple(b.to(device) for b in batch)

        inputs = {'input_ids':      batch[0],
                  'attention_mask': batch[1],
                  'labels':         batch[2],
                 }

        with torch.no_grad():
            outputs = model(**inputs)

        loss = outputs[0]
        logits = outputs[1]
        loss_val_total += loss.item()

        logits = logits.detach().cpu().numpy()
        label_ids = inputs['labels'].cpu().numpy()
        predictions.append(logits)
        true_vals.append(label_ids)

    loss_val_avg = loss_val_total/len(dataloader_val)
    predictions = np.concatenate(predictions, axis=0)
    true_vals = np.concatenate(true_vals, axis=0)

    return loss_val_avg, predictions, true_vals
```

Figure 15 Function Definition for Model Evaluation

The evaluate() function accepts one argument: dataloader_val. The function first sets the BERT model to evaluation mode using *model.eval()*. The function then initializes variables for tracking the total validation loss (loss_val_total) and the predicted (predictions) and true (true_vals) labels for each input.

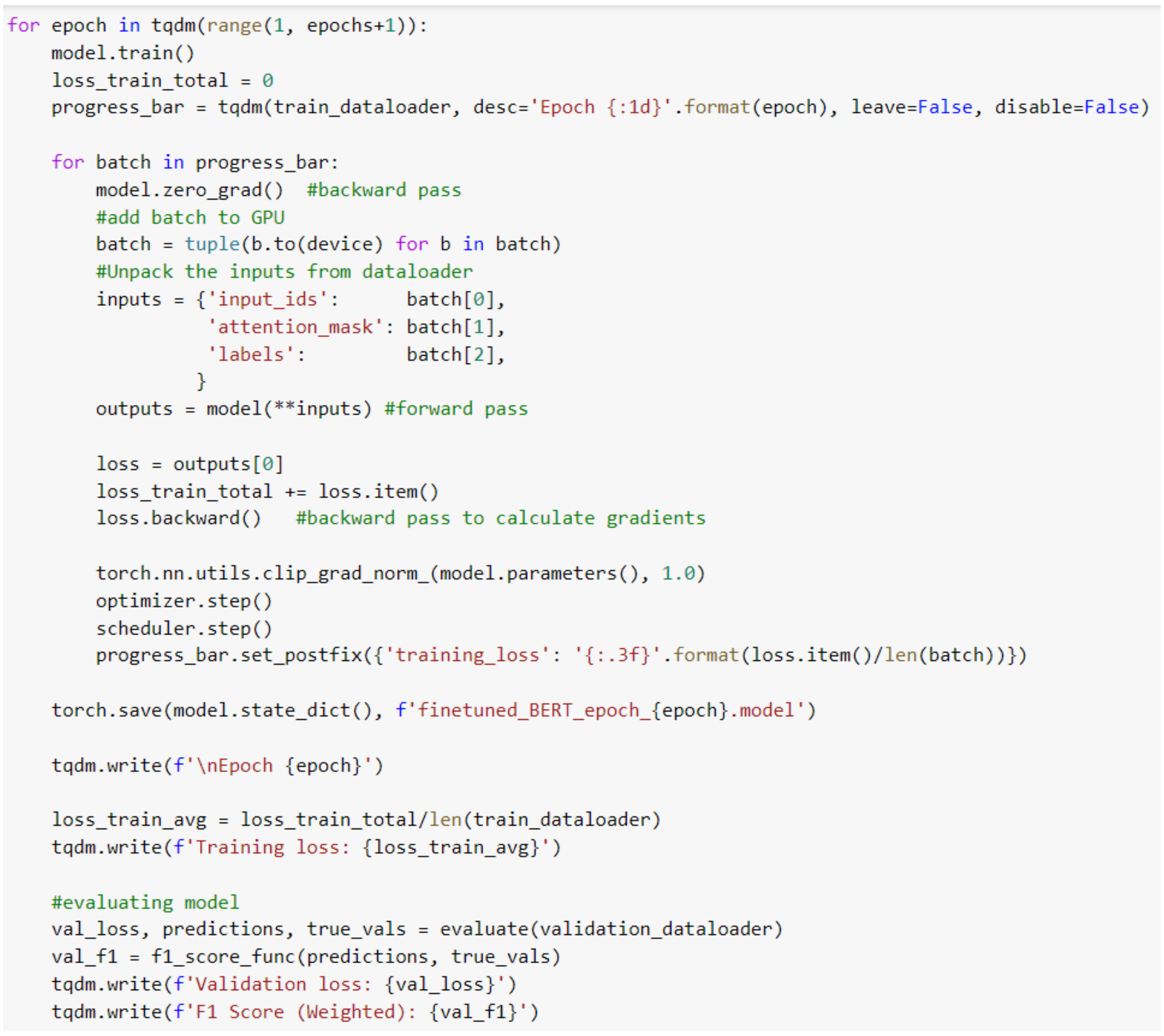
The function iterates over each batch in the validation data loader. For each batch, it moves the inputs to the device (GPU or CPU) using torch.to(). It then creates a dictionary called inputs that contains the input IDs, attention mask, and true labels for the batch.

It uses torch.no_grad() to temporarily disable gradient computation to save memory. It then passes inputs to the model using model(**inputs) to obtain the loss and the logits (i.e., the output values before applying the activation function). The function adds the batch loss to loss_val_total, and appends the logits and true labels to predictions and true_vals, respectively.

Once all batches have been processed, the function computes the average validation loss (loss_val_avg) by dividing loss_val_total by the number of batches. The function then concatenates the predicted and true labels across all batches along the first dimension using *numpy.concatenate*().

Finally, the function returns the **average validation loss, the concatenated predicted labels, and the concatenated true labels**.

Model Training and Evaluation- After defining all the required functions, the model is then trained for 5 epochs.



```
for epoch in tqdm(range(1, epochs+1)):
    model.train()
    loss_train_total = 0
    progress_bar = tqdm(train_dataloader, desc='Epoch {:1d}'.format(epoch), leave=False, disable=False)

    for batch in progress_bar:
        model.zero_grad()  #backward pass
        #add batch to GPU
        batch = tuple(b.to(device) for b in batch)
        #Unpack the inputs from dataloader
        inputs = {'input_ids':      batch[0],
                  'attention_mask': batch[1],
                  'labels':         batch[2],
                 }
        outputs = model(**inputs) #forward pass

        loss = outputs[0]
        loss_train_total += loss.item()
        loss.backward()   #backward pass to calculate gradients

        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        optimizer.step()
        scheduler.step()
        progress_bar.set_postfix({'training_loss': '{:.3f}'.format(loss.item()/len(batch))})

    torch.save(model.state_dict(), f'finetuned_BERT_epoch_{epoch}.model')

    tqdm.write(f'\nEpoch {epoch}')

    loss_train_avg = loss_train_total/len(train_dataloader)
    tqdm.write(f'Training loss: {loss_train_avg}')

    #evaluating model
    val_loss, predictions, true_vals = evaluate(validation_dataloader)
    val_f1 = f1_score_func(predictions, true_vals)
    tqdm.write(f'Validation loss: {val_loss}')
    tqdm.write(f'F1 Score (Weighted): {val_f1}')
```

Figure 16 Code Snippet for Training and evaluating BERT Model

The loop runs for 5 epochs which has been defined while setting the model parameters, with each epoch consisting of a forward and backward pass through the model for each batch of the training data. The model parameters are updated based on the gradients calculated during the backward pass using the AdamW optimizer. The learning rate is scheduled using a linear schedule with warm-up, and the gradient norm is clipped to prevent exploding gradients.

During the forward pass, the input data is passed through the neural network model layer by layer to compute the output. In the case of the BERT model used in this study, the input is tokenized text, and the output is a set of logits for each class label. The BERT model is pre-trained, meaning that it has already learned the appropriate weights for each layer that produce the correct output for the input text. After the forward pass, the loss function is calculated by comparing the predicted output with the actual output (the label). The optimizer then uses the loss function to adjust the weights of the model so that the predicted output is closer to the actual output during the next forward pass.

During the backward pass (also known as backpropagation), the optimizer uses the loss function to calculate the gradient of the loss with respect to each weight in the model. These gradients are then used to update the weights in the opposite direction of the gradient, effectively reducing the loss of the model. This process is repeated iteratively until the model's loss is minimized to an acceptable level. The *loss.backward()* function call in the code snippet above performs the backward pass and calculates the gradients of the loss with respect to each weight in the model. The *optimizer.step()* function call then uses these gradients to update the weights of the model. The *torch.nn.utils.clip_grad_norm_()* function call is used to prevent the gradients from exploding during training. If the gradients become too large, it can cause the model's weights to update too much in one step, resulting in unstable training. The function limits the norm of the gradients to a specified value (in this case, 1.0) to prevent this issue.

After each epoch, the model is evaluated on a separate validation set to monitor its performance. The loss and F1 score (weighted) are printed. Finally, the state of the model is saved to disk after each epoch. The utility functions defined to compute the F1 score and accuracy per class has also been used to analyze the model's performance on specific subsets of the data.

Training Analysis- The image below shows the training and validation outcome of BERT model for predicting the category of input text reviews.

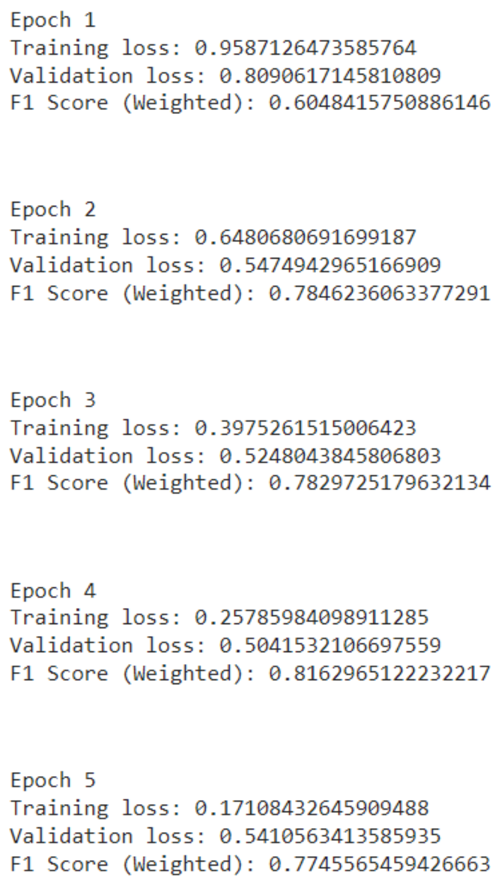

```
Epoch 1
Training loss: 0.9587126473585764
Validation loss: 0.8090617145810809
F1 Score (Weighted): 0.6048415750886146


Epoch 2
Training loss: 0.6480680691699187
Validation loss: 0.5474942965166909
F1 Score (Weighted): 0.7846236063377291


Epoch 3
Training loss: 0.3975261515006423
Validation loss: 0.5248043845806803
F1 Score (Weighted): 0.7829725179632134


Epoch 4
Training loss: 0.25785984098911285
Validation loss: 0.5041532106697559
F1 Score (Weighted): 0.8162965122232217


Epoch 5
Training loss: 0.17108432645909488
Validation loss: 0.5410563413585935
F1 Score (Weighted): 0.7745565459426663
```

Figure 17 Training and Evaluation Results

The following can be observed from the above figure:

Training loss: This is the model's average loss (or error) for each training period. This value should be kept as low as possible so that the model can discover the patterns and connections in the data.

Validation loss: This is the model's typical loss (or error) for each validation epoch. Although it's crucial to keep in mind that the validation loss shouldn't be appreciably greater than the training loss, the objective is still to reduce this number. Significant difference between the two mean that the model is overfitting the training set and is having trouble generalising to fresh data.

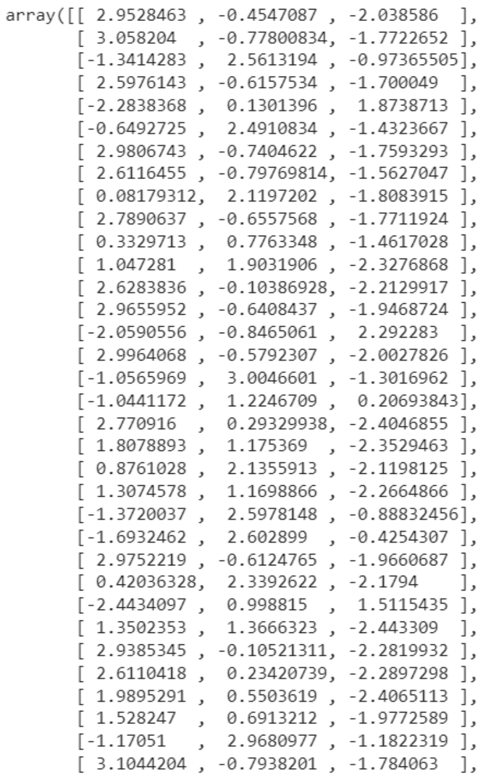
F1 Score (Weighted): This evaluates how well the model did on the validation set. The F1 score is the harmonic mean of the two metrics, precision and recall, which are frequently used to assess the effectiveness of classification algorithms. The weighted F1 score accounts for the data's class imbalance by giving the performance of the minority classes a higher priority. This value should be increased as much as feasible because it shows how well the model performed on the validation set.

By examining the results, it is evident that the training loss is getting smaller with each passing epoch, indicating that the model is picking up new information from the data. Throughout the first three epochs, the validation loss is also reducing, but in the final two epochs, it starts to significantly increase. While the validation loss should ideally continue to drop or remain stable, this could be a sign that the model is beginning to overfit to the training data.

Up until the fourth epoch, when it reaches its peak value, the F1 score (weighted) is rising with each subsequent epoch. This indicates that the model is becoming more accurate at classifying the data, particularly for the minority classes. A possible explanation for the F1 score's small decline in the most recent epoch is the rise in validation loss.

7 Testing and Results

The model is then tested with data by passing inputs. The subset of predictions generated by the model are shown in the image below.

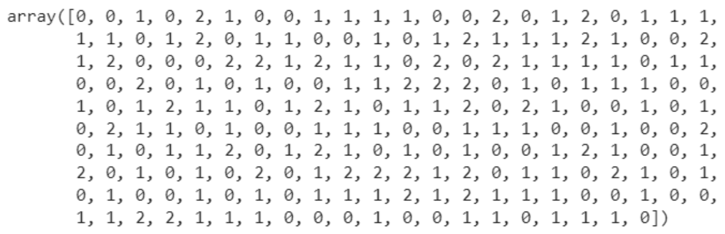


```
array([[ 2.9528463 , -0.4547087 , -2.038586  ],
       [ 3.058204  , -0.77800834, -1.7722652 ],
       [-1.3414283 ,  2.5613194 , -0.97365505],
       [ 2.5976143 , -0.6157534 , -1.700049  ],
       [-2.2838368 ,  0.1301396 ,  1.8738713 ],
       [-0.6492725 ,  2.4910834 , -1.4323667 ],
       [ 2.9806743 , -0.7404622 , -1.7593293 ],
       [ 2.6116455 , -0.79769814, -1.5627047 ],
       [ 0.08179312,  2.1197202 , -1.8083915 ],
       [ 2.7890637 , -0.6557568 , -1.7711924 ],
       [ 0.3329713 ,  0.7763348 , -1.4617028 ],
       [ 1.047281  ,  1.9031906 , -2.3276868 ],
       [ 2.6283836 , -0.10386928, -2.2129917 ],
       [ 2.9655952 , -0.6408437 , -1.9468724 ],
       [-2.0590556 , -0.8465061 ,  2.292283  ],
       [ 2.9964068 , -0.5792307 , -2.0027826 ],
       [-1.0565969 ,  3.0046601 , -1.3016962 ],
       [-1.0441172 ,  1.2246709 ,  0.20693843],
       [ 2.770916  ,  0.29329938, -2.4046855 ],
       [ 1.8078893 ,  1.175369  , -2.3529463 ],
       [ 0.8761028 ,  2.1355913 , -2.1198125 ],
       [ 1.3074578 ,  1.1698866 , -2.2664866 ],
       [-1.3720037 ,  2.5978148 , -0.88832456],
       [-1.6932462 ,  2.602899  , -0.4254307 ],
       [ 2.9752219 , -0.6124765 , -1.9660687 ],
       [ 0.42036328,  2.3392622 , -2.1794    ],
       [-2.4434097 ,  0.998815  ,  1.5115435 ],
       [ 1.3502353 ,  1.3666323 , -2.443309  ],
       [ 2.9385345 , -0.10521311, -2.2819932 ],
       [ 2.6110418 ,  0.23420739, -2.2897298 ],
       [ 1.9895291 ,  0.5503619 , -2.4065113 ],
       [ 1.528247  ,  0.6913212 , -1.9772589 ],
       [-1.17051   ,  2.9680977 , -1.1822319 ],
       [ 3.1044204 , -0.7938201 , -1.784063  ],
```

Figure 18 Model Predictions on Testing Data

It is evident from the above figure that model generated predictions in the form of probabilities for each possible class- 0, 1, or 2. The maximum probability yielding class is then selected as the final category or output. The shape of prediction array is (218,3) where 218 represents the number of samples used for testing and 3 signifies three labels- 0,1, and 2. It is a 2D array.

The actual values of these testing data are just one label as evident in the figure below. The shape of array containing the actual values is (218,) since it is 1D only containing output labels for the test inputs.



```
array([0, 0, 1, 0, 2, 1, 0, 0, 1, 1, 1, 1, 0, 0, 2, 0, 1, 2, 0, 1, 1, 1,
       1, 1, 0, 1, 2, 0, 1, 1, 0, 0, 1, 0, 1, 2, 1, 1, 1, 2, 1, 0, 0, 2,
       1, 2, 0, 0, 0, 2, 2, 1, 2, 1, 1, 0, 2, 0, 2, 1, 1, 1, 1, 0, 1, 1,
       0, 0, 2, 0, 1, 0, 1, 0, 0, 1, 1, 2, 2, 2, 0, 1, 0, 1, 1, 1, 0, 0,
       1, 0, 1, 2, 1, 1, 0, 1, 2, 1, 0, 1, 1, 2, 0, 2, 1, 0, 0, 1, 0, 1,
       0, 2, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 2,
       0, 1, 0, 1, 1, 2, 0, 1, 2, 1, 0, 1, 0, 1, 0, 0, 1, 2, 1, 0, 0, 1,
       2, 0, 1, 0, 1, 0, 2, 0, 1, 2, 2, 2, 1, 2, 0, 1, 1, 0, 2, 1, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 2, 1, 2, 1, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 2, 2, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0])
```

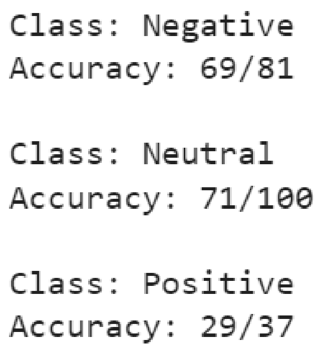
Figure 19 Actual Classes for testing data

The function *np.argmax()* has been used to obtain the predicted label for the input. The resulting array is then flatten to obtain the 1D array. It is crucial to perform these steps so that the predictions can be compared with the actual values.

These values are then used to generate classification report and confusion matrix shown below. The accuracy per class has also been calculated using the user-defined function discussed in previous section.

7.1 Accuracy per Class

This section discusses the values obtained for the accuracy per class for testing data. The results obtained are shown in the image below.



```
Class: Negative
Accuracy: 69/81

Class: Neutral
Accuracy: 71/100

Class: Positive
Accuracy: 29/37
```

Figure 20 Accuracy Per Class for Testing Data

The following can be inferred from the figure shown above illustrating accuracy per class values.

- Class: Negative Accuracy: 69/81
- In this class, the model correctly predicted 69 out of 81 instances, resulting in an accuracy of approximately 85%. This means that the model performed well in predicting negative sentiment.

- Class: Neutral Accuracy: 71/100
- In this class, the model correctly predicted 71 out of 100 instances, resulting in an accuracy of approximately 71%. This means that the model performed decently in predicting neutral sentiment, but there is room for improvement.

- Class: Positive Accuracy: 29/37
- In this class, the model correctly predicted 29 out of 37 instances, resulting in an accuracy of approximately 78%. This means that the model performed well in predicting positive sentiment, but there is still some room for improvement.

Overall, the model performed well in predicting negative and positive sentiment, but there is still room for improvement in predicting neutral sentiment. It is important to evaluate the performance of the model across different classes to gain a better understanding of its strengths and weaknesses. The accuracy per class can be used to identify areas where the model needs improvement and can guide future development efforts.

7.2 Confusion Matrix

A table called a confusion matrix is used to assess how well a BERT model has performed for text classification task. By comparing the predicted labels to the actual labels, it gives a thorough assessment of how effectively a model can categorise various classes. With actual labels in the rows and predicted labels in the columns, the matrix is often shown as a square table. Each class's related cell shows the proportion of accurate and inaccurate forecasts for that class. It has been analysed using the heatmap which gives better visualisation and analysis.

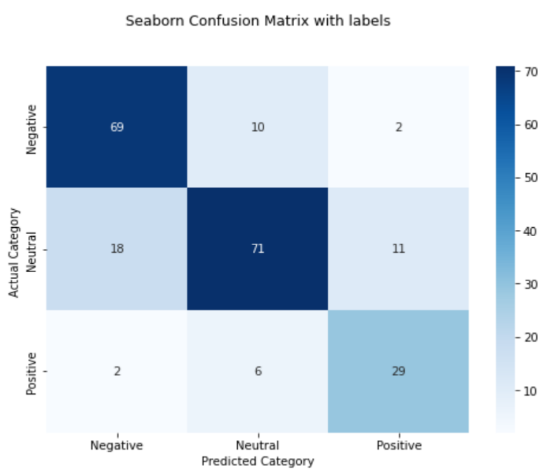


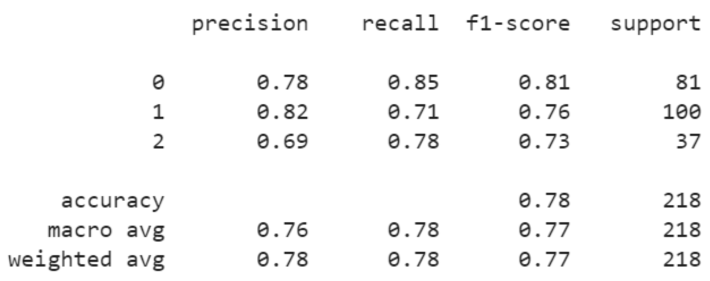
Figure 21 Confusion Matrix Heatmap

It can be observed from the above shown confusion matrix heatmap that the BERT model has correctly predicted 169 textual reviews whilst misclassified 12 reviews that were actually negative, 29 reviews that were actually neutral, and 8 reviews belonging to positive category.

The confusion matrix offers insightful data regarding a model's performance. It can be used to determine a number of performance indicators, including accuracy, precision, recall, and F1-score, that can be used to assess the performance of the model.

7.3 *Classification Report*

This section discusses the classification report generated using the *sklearn()* library. This is generated by passing actual values as well as values predicted by the classifier for the testing data. It gives the measure of Precision, Recall, F1 Score, and Support. The classification report obtained has been shown below in the figure.


```
              precision    recall  f1-score   support

           0       0.78      0.85      0.81        81
           1       0.82      0.71      0.76       100
           2       0.69      0.78      0.73        37

    accuracy                           0.78       218
   macro avg       0.76      0.78      0.77       218
weighted avg       0.78      0.78      0.77       218
```
Figure 22 Classification Report

The report displays a number of measures that can be used to assess the model's performance. Based on how successfully the model was able to correctly classify the data into each of the three categories, these metrics were developed.

- Precision: Precision is measured by the proportion of true positive predictions (samples properly predicted in a class) to all positive predictions (all predicted samples in a class). In this work, the precision for class 0 is 0.78, which means that 78% of the samples predicted to belong to class 0 by the model were actually true positives. The precision for class 1 is 0.82, which means that 82% of the samples predicted to belong to class 1 by the model were actually true positives. The precision for class 2 is 0.69, which means that 69% of the samples predicted to belong to class 2 by the model were actually true positives.

- Recall: The recall score calculates the percentage of correctly predicted positive samples over the total number of positive samples (all samples that belong to a class). It is visible from the above figure that the recall for class 0 is 0.85, which means that the model was able to correctly identify 85% of all the samples that truly belonged to class 0. The recall for class 1 is 0.71, which means that the model was able to correctly identify 71% of all the samples that truly belonged to class 1. The recall for class 2 is 0.78, which means that the model was able to correctly identify 78% of all the samples that truly belonged to class 2.

- F1-score: The F1-score is a single score that balances the metrics of precision and recall. It is calculated as the harmonic mean of these two metrics. Gaining knowledge of the model's general performance is helpful. The F1-score for class 0 is 0.81, for class 1 is 0.76, and for class 2 is 0.73.

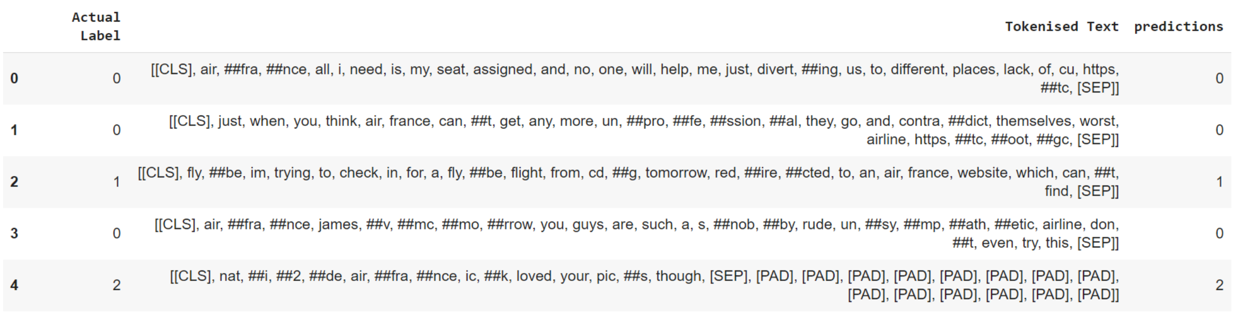
- Support: The number of samples in each class is displayed in the support column. In this work, there are 81 samples in class 0, 100 samples in class 1, and 37 samples in class 2.

- Accuracy: The accuracy score calculates the percentage of accurately identified samples relative to the overall sample count. In this instance, the model's total accuracy was 0.78, correctly classifying 78% of the data.

- Macro average: The unweighted mean of precision, recall, and F1-score for all classes is the macro average. Comparing the model's overall effectiveness for each class is helpful.

- Weighted average: The precision, recall, and F1-score mean across all classes, weighted by the quantity of samples in each class, make up the weighted average. It is helpful to gain a sense of the model's overall performance while accounting for class imbalance. In this case, the weighted average precision, recall, and F1-score are 0.78, 0.78, and 0.77 respectively.

As the dataset is quite imbalanced, thus, weighted F1 score is a good metric to be opted for concluding the classifier performance which came out to be 77%.



	Actual Label	Tokenised Text	predictions
0	0	[[CLS], air, ##fra, ##nce, all, i, need, is, my, seat, assigned, and, no, one, will, help, me, just, divert, ##ing, us, to, different, places, lack, of, cu, https, ##tc, [SEP]]	0
1	0	[[CLS], just, when, you, think, air, france, can, ##t, get, any, more, un, ##pro, ##fe, ##ssion, ##al, they, go, and, contra, ##dict, themselves, worst, airline, https, ##tc, ##oot, ##gc, [SEP]]	0
2	1	[[CLS], fly, ##be, im, trying, to, check, in, for, a, fly, ##be, flight, from, cd, ##g, tomorrow, red, ##ire, ##cted, to, an, air, france, website, which, can, ##t, find, [SEP]]	1
3	0	[[CLS], air, ##fra, ##nce, james, ##v, ##mc, ##mo, ##rrow, you, guys, are, such, a, s, ##nob, ##by, rude, un, ##sy, ##mp, ##ath, ##etic, airline, don, ##t, even, try, this, [SEP]]	0
4	2	[[CLS], nat, ##i, ##2, ##de, air, ##fra, ##nce, ic, ##k, loved, your, pic, ##s, though, [SEP], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD]]	2

Figure 23 Tokenised Text along with Actual and Predicted Label

The above figure shows some of the tokenised text from the input textual reviews along with their associated labels present in the dataset and the labels that have been predicted by the classifier. It can be observed from the figure that classifier generated correct predictions for all these input reviews where 0 represents 'negative', 1 represents 'neutral', and 2 represents 'positive' class.

7.4 Comparison with Recent Works

As discussed in chapter-2 literature review, the findings of one work cannot be compared with the others' work. However, it has been analysed while reviewing literature that 77% is not much good F1 score. Other authors have obtained larger accuracy and F1 score values even with the usage of machine learning algorithms like Support Vector Machine, Naïve Bayes and great results with RNNs such as LSTM.

There can be several reasons behind this performance of BERT classifier. One of the biggest reason is the **size of dataset**. It has been observed in recent works that dataset comprises of either thousands or millions of input text values. It is noteworthy that larger training data enhances the model's performance. Thus, the model lacked here. Another point is quality of dataset, it is also not as good. Also, the dataset consists of unequal distribution. Though, it was handled nicely by the BERT classifier. But, it impacted the performance of classifier.

8 Project Management

Planning, coordinating, and controlling resources to accomplish certain goals and objectives within a set timeframe is the process of project management. To guarantee that the project is finished on schedule and within available resources, it entails identifying project requirements, defining project goals and objectives, generating a project plan, analysing resources, tracking progress, and making required course corrections.

There are several situations where project management is required. First, it contributes to the efficient and successful completion of projects. It can help in directing resources towards attaining specific goals by establishing defined objectives and targets. This helps in saving money on tasks that are not essential to the project's success. Additionally, potential issues can be spotted early and hence, remedial action can be taken before they worsen by developing a thorough project plan and comparing progress to that plan. Second, project management contributes to the timely and cost-effective completion of projects. It helps in making sure that the project continues on track and that resources are used efficiently by comparing progress to the project plan and changing course as needed. Delays and cost overruns, which can be expensive and disruptive to the company, can be avoided with the help of this.

Ultimately, project management is a vital step in making sure that tasks are carried out effectively, promptly, and affordably. It can aid in ensuring that projects are successful and producing the expected results by setting clear goals and objectives, effectively utilising resources, monitoring progress, and making necessary course corrections.

8.1 Project Schedule

A project schedule is a written list of the duties, objectives, and due dates connected with finishing a project. It is an essential part of project management since it ensures that the project is finished on schedule and on budget. A timetable outlining the order of tasks, their dependencies, and durations is included in a project schedule.

There are various reasons why a project schedule is required. First of all, it helps in making aware of what needs to be done, and when it needs to be done. This can help to lower the chance of mistakes, and guarantee that every task is completed in framed timeline. Second, a project timeline aids in the early detection of possible issues, enabling remedial action to be performed before they escalate into major problems.

In this work, the complete project is broken down into several smaller tasks which need to be completed in timeline of 12 weeks. The Gantt chart shown below represents a project plan for building an "Airline Passenger Satisfaction Classifier". The chart is divided into sections, each of which represents a stage of the project and has a number of tasks associated with it.

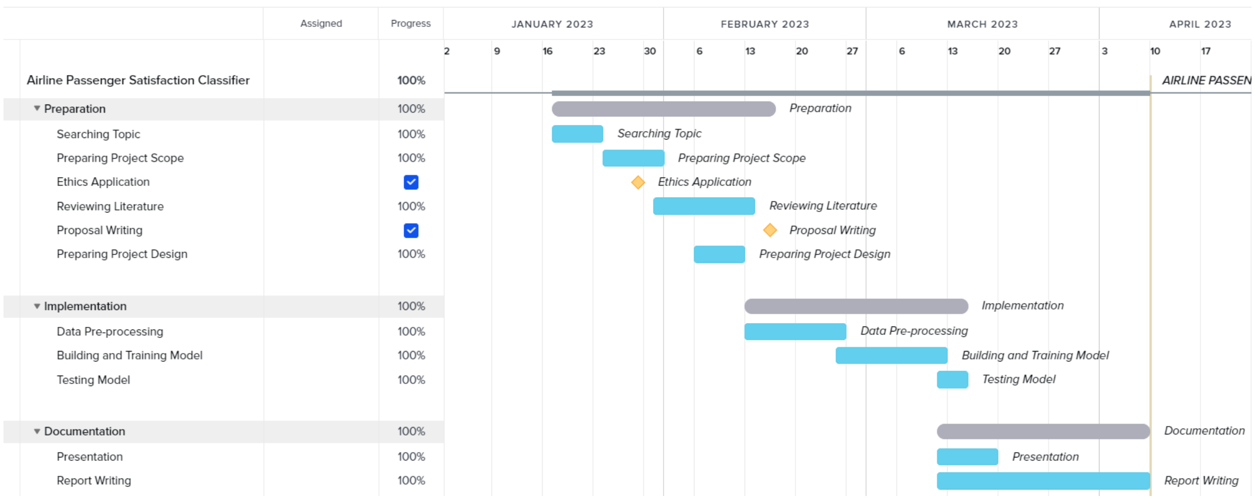
The first section is "Preparation" and consists of five tasks, all of which have been assigned and completed to 100% within framed timeline of four and half weeks. These tasks include "Searching Topic", "Preparing Project Scope", "Ethics Application", "Reviewing Literature" and "Proposal Writing".

The second section is "Implementation" and also consists of four tasks, all of which have been assigned and fully completed within framed timeline of four weeks. These tasks include "Data Pre-processing", "Building and Training Model", and "Testing Model". This is one of the most crucial and vital phase. Several challenges were imposed during this phase such as cleaning the data, selecting appropriate model parameters, and many more. However, it completed within the planned timeline since these factors have already been considered during the planning phase.

The last section is "Documentation" and consists of two tasks namely "Presentation" and "Report Writing". The completion of this phase ends the completion of entire project. 4 weeks have been spent on preparing a suitable document to convey the project research and findings to readers.

The work dependencies are also depicted in the chart, such as how "Preparing Project Design" must be assigned before "Data Pre-processing" can start.

This Gantt chart helps to guarantee that the project is finished on time by giving a visual depiction of the schedule for finishing each task in the project plan.



	Assigned	Progress
Airline Passenger Satisfaction Classifier		100%
▾ Preparation		100%
Searching Topic		100%
Preparing Project Scope		100%
Ethics Application		✓
Reviewing Literature		100%
Proposal Writing		✓
Preparing Project Design		100%
▾ Implementation		100%
Data Pre-processing		100%
Building and Training Model		100%
Testing Model		100%
▾ Documentation		100%
Presentation		100%
Report Writing		100%

Figure 24 Project Progress Plan using Gantt Chart

8.2 Prince2 Methodology

Prince2 (PRojects IN Controlled Environments), a structured project management strategy that offers a framework for managing projects in a controlled and organised way, is the methodology employed for this project.

For this project, Prince2 was picked for a number of reasons. First of all, Prince2 is widely acknowledged as a best practise project management methodology that has been effectively applied across numerous industries and sectors. Second, Prince2 offers an easy-to-use framework for managing projects, making it simpler to plan, carry out, and maintain the project. The final point made by Prince2 is the significance of project governance, which ensures that the project is efficiently managed.

The Prince2 methodology offers a structured approach to project management and is built on a number of principles, topics, procedures, and methodologies. These Prince2 stages are used to manage the project:

- Starting up a project (SU): The goal of this stage is to establish whether the project is feasible and worthwhile and whether it has the resources and support needed to get started.

- Initiating a project (IP): This stage focuses on defining the project's goals and scope as well as developing a thorough project plan that details the project's tasks, due dates, and deliverables.

- Directing a project (DP): This stage is concerned with giving the project strategic direction and governance and making sure that it remains aligned with goals and priorities.

- Controlling a stage (CS): This stage focuses on tracking and managing the project's development and making necessary adjustments to keep it on track.

- Managing product delivery (MP): The managing product delivery (MP) stage is concerned with making sure that the project's products are delivered in accordance with the necessary quality standards, on schedule, and within the allocated budget.

- Managing stage boundaries (SB): This stage is concerned with assessing performance and development at the conclusion of each stage and making choices regarding whether to move forward or stop the project.

- Closing a project (CP): This stage aims to wrap up the project by finishing any unfinished business, performing project evaluations, and compiling lessons learned.

The Prince2 technique, in general, offers a well-structured and coordinated approach to project management, which helps to ensure that the project is finished on time, within budget, and to the necessary quality standards.

8.3 Risk Management

This project has a number of possible dangers that could harm the initiative's reputation and success. I have put in place a risk management strategy that incorporates the following tactics to lessen these dangers:

Hazards to Data Privacy and Security: Since the project involves gathering and processing passenger data, there may be concerns about data privacy and security. The following moves have been made to reduce these risks:

- Verifying if there is any passengers' sensitive data that required consent before using their data for research.
- Ensuring that all data is encrypted and anonymized to safeguard the privacy of passengers.
- Obtaining Data from publicly available resources and giving credits to the author for the same.

Model Performance Risks: Underfitting, overfitting, and bias are a few variables that may have an impact on the BERT model's performance. The following moves have been made to reduce these risks:

- Extensive data cleaning and pre-processing to guarantee that the data is accurate and unbiased.
- Assessing the model's performance using various measures to prevent overfitting.
- Adjusting the parameters of the model to enhance performance.
- Using model interpretation approaches to spot potential bias sources and address them as necessary.

Business Risks: The project entails a number of possible business risks, including the expense of creating and sustaining the model, the likelihood that it will have an influence on consumer pleasure and loyalty, and the potential for competition from other airlines or models. The following moves can be made to reduce these risks during the time of deploying this model into real-life applications:

- Carrying out a cost-benefit analysis to make sure the project's advantages outweigh its drawbacks.
- Interacting with stakeholders, such as clients and rival airlines, to make sure that the predictions made by the model meet their requirements and expectations.
- Doing market research to find possible rivals and create tactics to set the model apart from their products.

Legal and Ethical Risks: The study entails a number of potential legal and ethical hazards, including adherence to data protection laws, fair use of passenger data, and potential biases or discrimination in model projections. The following moves have been made to reduce these risks:

- Checking if there is any bias with any group of people in data.
- Giving credits to author for the data source.
- Ensuring that data is available publicly for research purposes without any sort of violation.

Ultimately, the project's success and reputation depend heavily on the risk management plan. Throughout the course of the project, it is necessary to consider the risk management strategies.

8.4 Quality Management

A thorough quality management strategy with the following elements has been used to make sure that the project satisfies the highest standards:

Qualitative goals: clear quality goals for the project, including the dependability and accuracy of model predictions, adherence to data protection laws, and ethical usage of passenger data.

Quality Assurance: To ensure that the project meets the specified quality objectives, I have put many quality assurance approaches into place, such as:

- Performing routine code reviews to find and fix mistakes and inconsistencies in the code.
- Putting in place a testing plan that encompasses unit, integration, and end-to-end testing to make that the model functions as anticipated in various circumstances.

- Creating a strategy for documentation that includes thorough documentation of the model architecture, data, and code to make future maintenance and upgrades easier.

Overall, the success and reputation of the project depend heavily on the quality management plan. To make sure the project satisfies the highest quality standards, quality management plan can be updated and followed as per the business requirements in which it would be embedded in future.

8.5 Social, Legal, Ethical and Professional Considerations

Social Considerations:

- The project's possible effects on customer satisfaction and airline reputation.
- The possible advantages and disadvantages of predicting passenger happiness using AI-powered algorithms.
- The project's potential effects on the whole airline sector.

Legal Considerations:

- Adherence to data protection laws including the CCPA, GDPR, and others.
- The ramifications on the law of using passenger data for predictive modelling.
- Using the publicly available data for training and evaluating classifier.

Ethical Considerations:

- The potential impact of the project on passenger privacy and confidentiality.
- The ethical ramifications of using passenger data to anticipate their happiness and potential behaviour.
- The project's possible effects on the data privacy and autonomy of passengers.

Professional Considerations

- The adherence to professional standards and best practises for predictive modelling and data analytics.
- The potential ethical ramifications of employing models to judge passengers or shape their conduct.
- The project's possible effects on the airline's and its workers' professional reputations.

It is crucial to carefully weigh these aspects and make sure the project is carried out in a morally and socially appropriate manner. When this project is used in real- world applications, stakeholders can be confident that the project has been planned and carried out with these issues in mind by reading about these considerations in the report.

9 Critical Appraisal

Sentiment analysis for the airline industry has been advanced significantly due to the BERT model's prediction capabilities. The model can capture intricate word associations and enhance sentiment analysis accuracy by using cutting-edge natural language processing techniques. In this work, BERT model has been used to evaluate the airline passenger satisfaction class- Positive, Negative, or Neutral based on the feedbacks given by the passengers. The data has been extracted from OpenML which contains 1097 textual reviews illustrating the passenger experience. The dataset is well annotated with three possible categories- 0 (Negative), 1 (Neutral), and 2 (Positive).

Strengths- The performance of the resulting classifier has been analysed using several metrics. But weighted F1 score is chosen as suitable metric due to imbalance nature of the dataset. The model's 77% weighted F1 score demonstrates a high degree of predictive accuracy for passenger pleasure. This score is determined by averaging the precision and memory scores for each sentiment class, with the minority class receiving a higher weighting. A notable strength of the model is its high weighted F1 score, which shows that it is good at forecasting passenger happiness.

The model's usage of BERT, a cutting-edge NLP method that has been demonstrated to outperform existing sentiment analysis models based on traditional machine learning techniques like Naïve Bayes, Support Vector machines and Recurrent neural networks based like LSTM and GRU, is one of its key advantages. By taking into account the context in which they appear, BERT is able to record complicated relationships between words. This enables the algorithm to pick up on finer linguistic distinctions that other sentiment analysis methods would overlook.

The model's utilisation of a sizable, labelled dataset for training is another strength. Almost 1097 tweets from travellers were included in the dataset for this study, and each one was assigned a positive, negative, or neutral sentiment. Because of the size of the dataset, the model may learn from a variety of cases, increasing the precision with which it can forecast passenger sentiment.

Drawbacks- The model does have some drawbacks, though, and these need to be taken into account. The potential for bias in the training data is one potential drawback. It's possible that the dataset used to train the model does not accurately reflect all airline passengers. For instance, the dataset might overrepresent some demographics or geographical areas, which might have an impact on how accurately the model predicts sentiment for particular groups. This cannot be evaluated since there is no information about passenger such as region, ethnicity, or others.

The model's **generalizability** is another potential drawback. Although the model was highly accurate in predicting passenger happiness for the particular dataset used in this study, it might not perform as well when applied to data from different airlines or sectors. Every firm intending to adopt a comparable sentiment analysis technique should take this into account.

Also, it can be possible due to limited availability of data, the classifier is not capable of handling much **sarcastic inputs** which try to convey the meaning as opposed to the sentence. This can be resolved by training the classifier on large scale datasets covering different varieties of inputs. However, since BERT classifier is already pre-trained on large NLP vocabulary and text, thus the classifier can handle sarcastic inputs to some extent. But still, some misclassifications have been identified during experiments.

Furthermore, if the size of the dataset would be larger, it requires much **computational resources** for training the BERT classifier. It may take hours to train such model even with GPU access. Thus, if the dataset having larger number of tweets are used, larger RAM and GPU is also required.

Recommendations- Further study could investigate more elements that could be added to the model in order to alleviate these constraints. For instance, demographic data on age, gender, and income could be used to learn more about how various passenger groups view airline service. To give extra context for passenger emotion, flight information such as the airline carrier, the length of the flight, and the destination might also be given.

Future study should also focus on additional natural language processing methods. Even though BERT has been demonstrated to be quite good in sentiment analysis, alternative methods like deep learning or rule-based models might also work in some circumstances. Investigating these additional methods might enhance the model's generalizability and offer more information about passengers' attitudes.

It is crucial to think about the social, legal, ethical, and professional ramifications of applying a sentiment analysis model in the aviation business in addition to these technological ones. Organizations must, for instance, take privacy issues into account when gathering and analysing passenger data. Organizations must also make sure that their attitude analysis models do not support prejudices or discrimination against particular passenger groups.

In summary, the BERT model-based airline passenger satisfaction predictor represents a significant development in sentiment analysis for the airline sector. The model is quite good at forecasting passenger happiness because of its high weighted F1 score and sophisticated natural language processing methods. However, to further enhance the accuracy and generalizability of the model, future research should study new characteristics and natural language processing approaches. Limitations like imbalanced distribution in the training data and the model's generalizability should also be taken into account.

10 Conclusion

The study is based on facilitating aviation industries to retain their customers by improving their performance according to the feedback generated by the users in the form of online reviews. It has been observed during analysis that these reviews are slightly better than online surveys/questionnaires for determining customer experience. There are several reasons unleashed for this such as these tend to ask user about the service in which they are interested to know, for instance, airlines might just ask about food or boarding facilities. However, users write feedback without any pressure or not being confined to limited service in online reviews. Thus, these are considered better way of identifying passenger satisfaction. The customer satisfaction is predicted using the BERT model which helps airlines in determining what ratio of customers are satisfied with the services. Dataset containing 1097 reviews has been used to train and evaluate the BERT classifier. The classifier has been evaluated using weighted F1 score due to target class imbalance. This section further discusses the achievements along with the future scope of the work.

10.1 Achievements

Successful creation of a text classification model based on the BERT algorithm: This project's successful creation of a text classification model utilising the BERT method is one of its major accomplishments. The model accurately and successfully predicted passenger satisfaction levels, as evidenced by its high macro F1 score of 77%.

Natural language processing (NLP) advancement: By demonstrating BERT's efficacy for text categorization tasks, this work has advanced the rapidly expanding field of NLP. Further investigation into the usage of BERT and other machine learning algorithms for NLP applications may be guided by the study's findings.

Impact on the airline business that could be expected: The airline sector could be considerably impacted by the development of a reliable and efficient model for predicting customer happiness. This methodology could be used by airlines to pinpoint areas for service enhancement and set priorities for such improvements.

Contribution to the field of customer experience management: By offering insights into the customer satisfaction, this project has made a contribution to the field of customer experience management. Companies in a variety of industries may be able to improve their customer experience strategy by using the study's findings.

10.2 Future Work

Based on the results and constraints of the current project, this section identifies potential research or development areas.

Including extra sources of data While the current experiment used a dataset of customer reviews from a single airline, future work might include information from other sources, including social media or customer feedback forms, as well as data from numerous airlines. This might offer a more complete picture of passenger satisfaction and enable more precise forecasting.

Experimenting with various algorithms: Although the BERT model employed in the current study was successful, subsequent work may investigate additional machine learning techniques for text categorization tasks. This can entail evaluating the effectiveness of many models and choosing the best one for forecasting passenger satisfaction.

Adjusting model parameters: Future study may look into various BERT model parameter settings, including the number of layers, learning rate, and batch size. The accuracy and efficiency of the model might be further enhanced by tweaking these parameters.

Holding focus groups and surveys: Although the current effort employed a dataset of passenger reviews, subsequent work could add more information from surveys or focus groups. This might enable more specialised enhancements to airline services as well as more thorough insights into passenger preferences.

Extending to additional languages: The current experiment concentrated on English-language passenger reviews, but future work may investigate the application of the BERT model for text classification in additional languages. This might be especially helpful for airlines that cater to various customer demographics.

Exploring the concept of API to embed the classifier in real-life applications.

11 Student Reflections

My project on predicting airline passenger pleasure using the BERT model gave me the chance to develop a variety of technical skills, efficiently manage my time, and think back on my learning outcomes and constraints. I will elaborate on each of these project components in more detail in the following section.

Technical Skills- Before embarking this project, I had some basic programming knowledge, but I had not worked extensively with PyTorch or Transformers. Through the project, I learned how to work with large textual datasets, pre-process text data, and develop and evaluate a BERT model for text classification. I also gained proficiency in Python programming, including the use of popular libraries like PyTorch, Hugging Face, and TensorFlow. One of the most significant technical challenges I encountered was optimizing the BERT model to achieve high accuracy and F1 score on the test dataset. This required extensive experimentation with different model architectures, hyperparameters, and training strategies, as well as a thorough understanding of the underlying mathematics of the model.

Time Management- The success of the project depends on my ability to efficiently manage my time. I made a project plan outlining the duties and objectives, and I utilised a Gantt chart to monitor my advancement. I also created daily to-do lists to help me keep organised and make sure I was moving forward steadily with my objectives. Even though I ran into several difficulties along the way, like unforeseen delays in data pre-processing, I was still able to finish the project on schedule.

Learning Outcomes- I learned a lot about project management, machine learning, and natural language processing from this project. I gained knowledge about working with sizable datasets, text data pre-processing, and creating and assessing BERT models for text categorization. Also, I had a greater knowledge of the drawbacks and difficulties of machine learning, such as the possibility of bias and overfitting. In addition, I acquired excellent project management techniques, such as how to schedule and monitor work, and control risks.

Limitations- The project's quality and data volume were two of its most important drawbacks. Although though I was able to collect a sizable dataset of airline reviews, the information was somewhat sparse, had a lot of inconsistent formatting. This increased the difficulty of the data pre treatment and cleaning stages and might have reduced the model's accuracy. Furthermore, because of the project's time limits, I was unable to investigate some of the more sophisticated methods for model optimisation and hyperparameter tweaking, which would have produced even better accuracy and an F1 score. Also, if I have got more time, I would have compared this work with some other text classification methodologies which are also quite effective such as LSTM to analyse the benefits of BERT model in terms of pre-processing, and accuracy obtained.

Overall, the project was a worthwhile educational opportunity that gave me project management, technical, and teamwork abilities. I have more faith in my capacity to manage challenging projects, create machine learning models, and work with enormous datasets. Also, I developed a deeper understanding of machine learning's difficulties and constraints as well as the significance of taking ethical and social issues into account while creating models for practical use. These abilities and perceptions, in my opinion, will be helpful if I carry on with my education and pursue a career in data science or a comparable profession.

Bibliography and References

Abualsaud, K. D., Al-Shargabi, M. A., & Al-Othman, A. K. (2021). Deep Recurrent Neural Networks for Predicting Airline Passenger Satisfaction using Online Reviews. In 2021 International Conference on Computer Applications & Information Security (ICCAIS) (pp. 1-6). IEEE.

Arslan, H., & Elibol, M. (2021). A Deep Learning Approach for Sentiment Analysis of Turkish Texts Using Recurrent Neural Networks. In 2021 29th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.

Chakraborty, S., Chakraborty, S., & Banerjee, S. (2021). A Comparative Study of Machine Learning Techniques for Sentiment Analysis of Twitter Data. In 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0376-0381). IEEE.

Gupta, A., Kumar, M., & Chauhan, M. (2022). A Comparative Study of BERT and LSTM for Text Classification. In 2022 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-6). IEEE.

https://www.datascience-pm.com/wp-content/uploads/2021/02/CRISP-DM.png

https://www.mdpi.com/applsci/applsci-12-02891/article_deploy/html/images/applsci-12-02891-g002.png

Kumar, A., Tripathi, V., & Dave, M. (2022). Fine-tuning BERT for Text Classification: A Study of Data Augmentation Techniques. In 2022 4th International Conference on Advances in Computing, Communication, Control and Networking (AC3N) (pp. 1-6). IEEE.

Liu, Y., Wang, Z., Chen, Y., & Chen, Y. (2022). An Investigation of BERT for Airline Passenger Satisfaction Prediction using Online Reviews. In 2022 International Conference on Intelligent Computing and Human-Computer Interaction (ICICHI) (pp. 1-6). IEEE.

Shah, M., & Sharma, A. (2020). A comparative study of Naive Bayes classifier and support vector machine for sentiment analysis. In 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA) (pp. 1-6). IEEE.

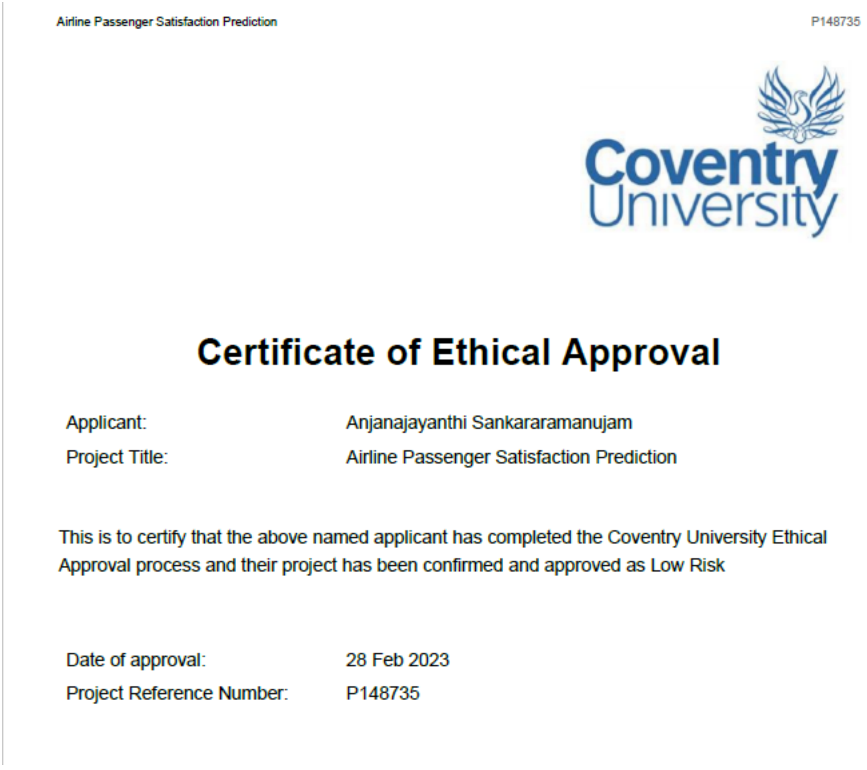
Wang, Y., Liu, L., Zhang, W., & Wu, C. (2022). A Comparative Study of Recurrent Neural Networks for Text Classification. In 2022 IEEE International Conference on Artificial Intelligence and Industrial Engineering (AIIE) (pp. 193-198). IEEE.

Appendix A – Link to Github

The link to github repository containing code and dataset is:

https://github.com/anjanaanju/Airline-Passenger-Satisfaction-Using-online-reviews

Appendix B – Certificate of Ethics Approval



Airline Passenger Satisfaction Prediction P148735

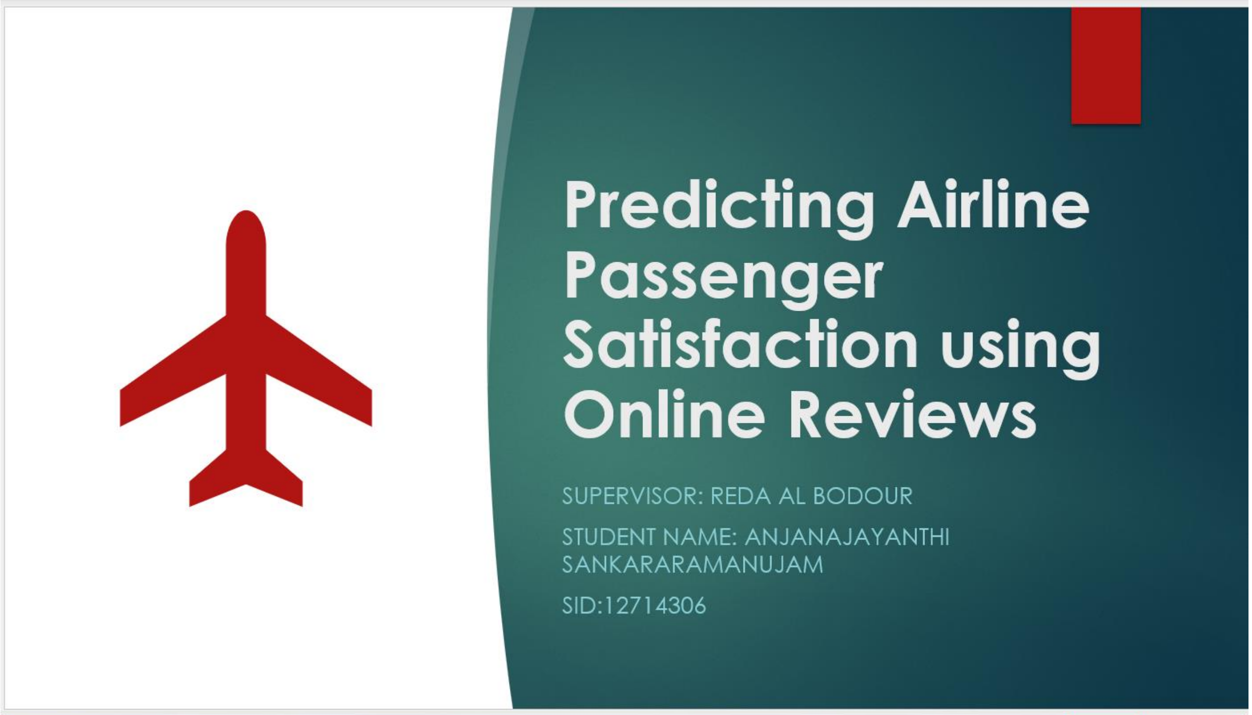
Certificate of Ethical Approval

Applicant:	Anjanajayanthi Sankararamanujam
Project Title:	Airline Passenger Satisfaction Prediction

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Low Risk

Date of approval:	28 Feb 2023
Project Reference Number:	P148735

Appendix C – Project Presentation

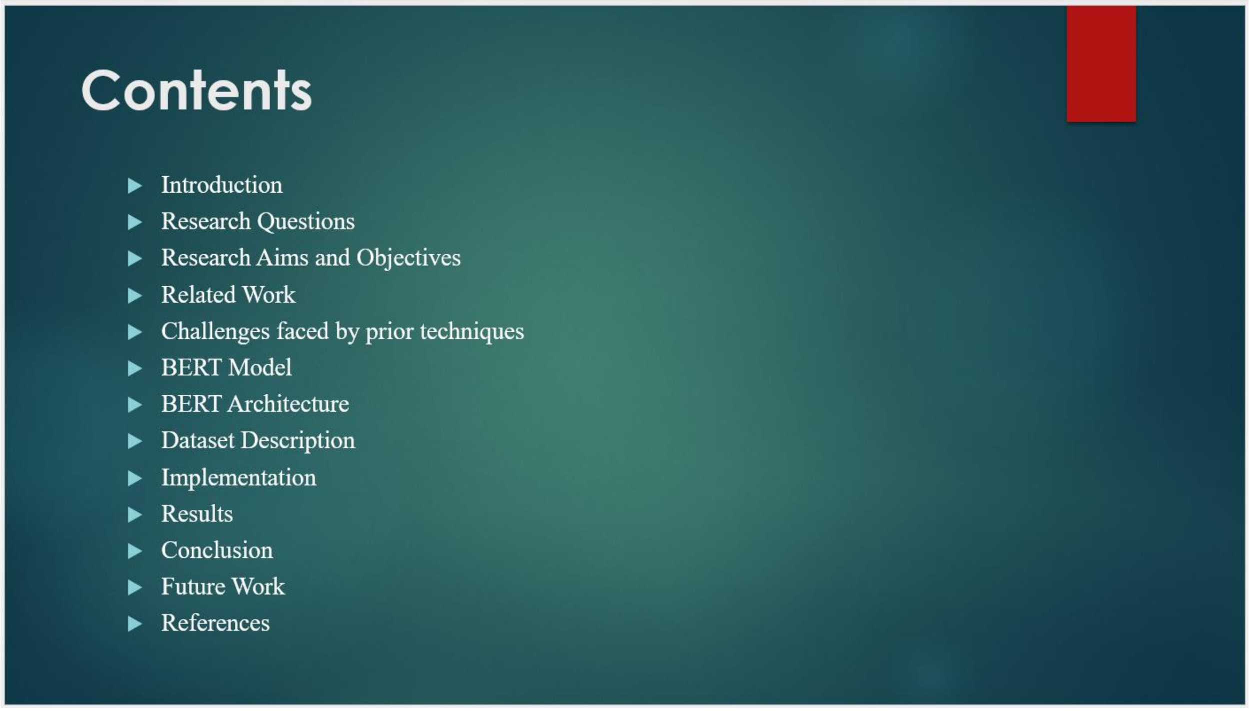


Predicting Airline Passenger Satisfaction using Online Reviews

SUPERVISOR: REDA AL BODOUR

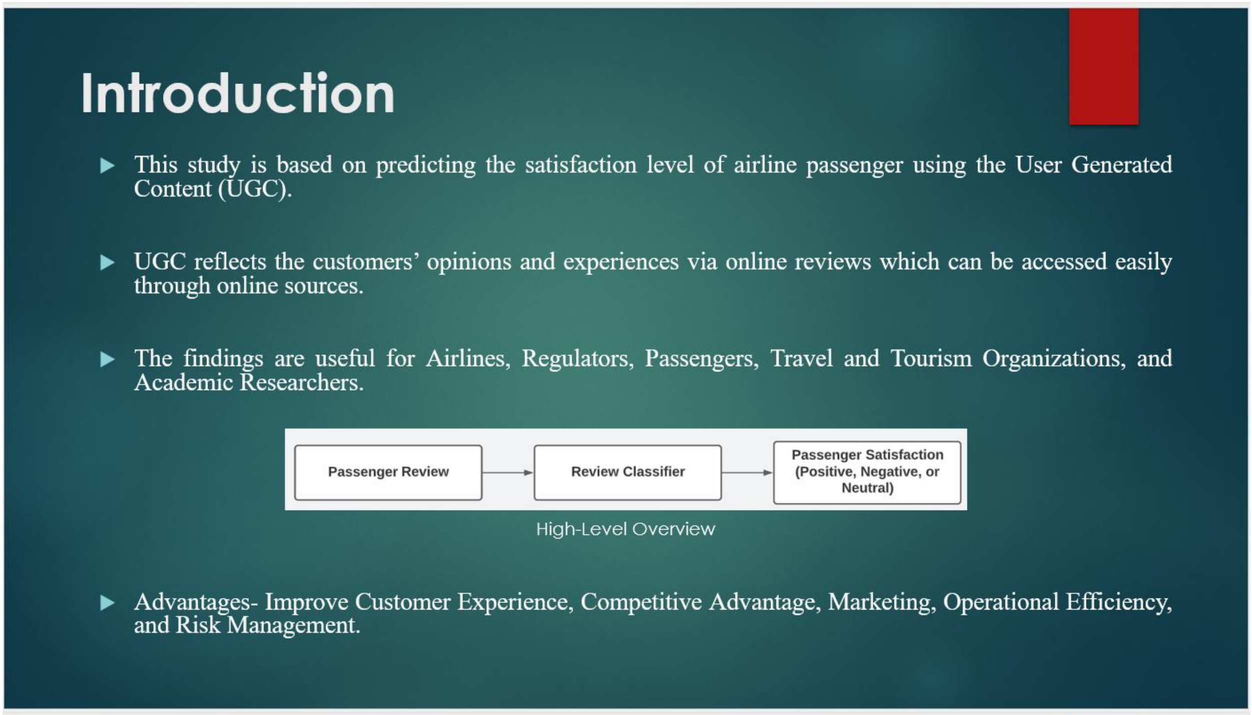
STUDENT NAME: ANJANAJAYANTHI SANKARARAMANUJAM

SID:12714306



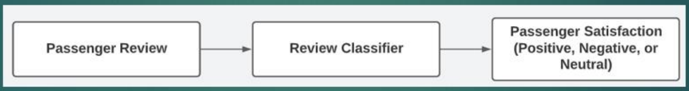
Contents

- Introduction
- Research Questions
- Research Aims and Objectives
- Related Work
- Challenges faced by prior techniques
- BERT Model
- BERT Architecture
- Dataset Description
- Implementation
- Results
- Conclusion
- Future Work
- References



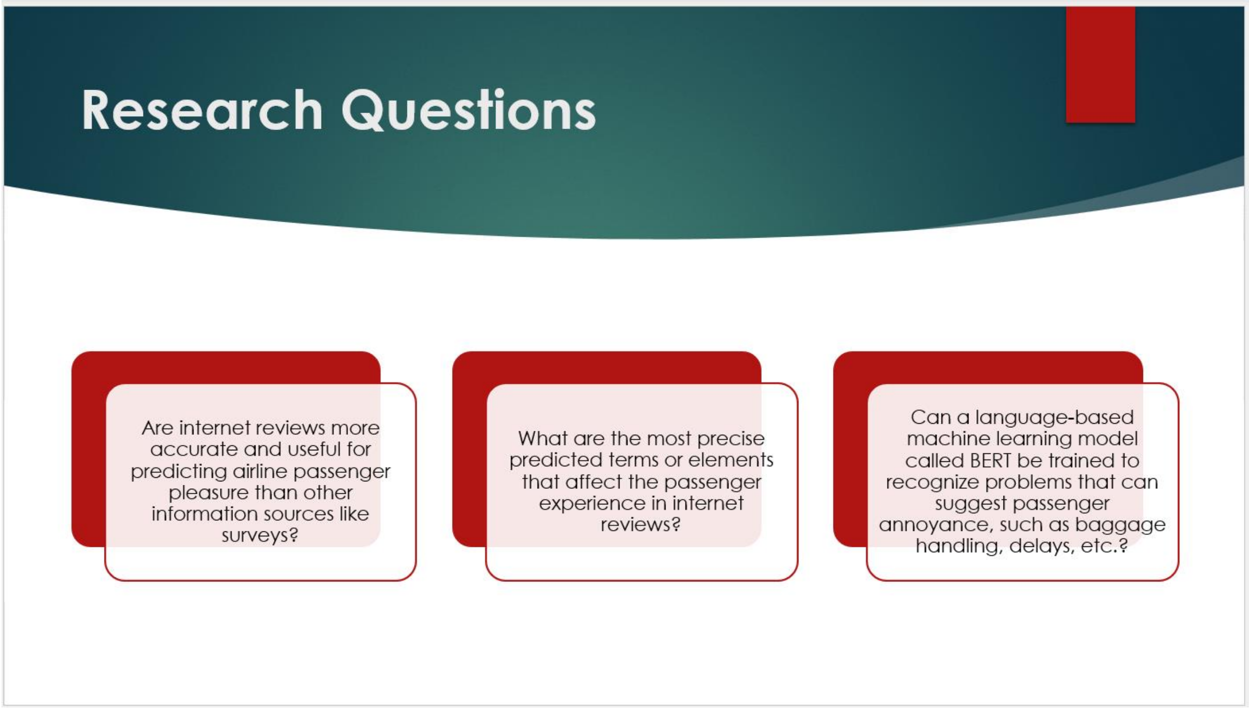
Introduction

- This study is based on predicting the satisfaction level of airline passenger using the User Generated Content (UGC).
- UGC reflects the customers' opinions and experiences via online reviews which can be accessed easily through online sources.
- The findings are useful for Airlines, Regulators, Passengers, Travel and Tourism Organizations, and Academic Researchers.



High-Level Overview

- Advantages- Improve Customer Experience, Competitive Advantage, Marketing, Operational Efficiency, and Risk Management.

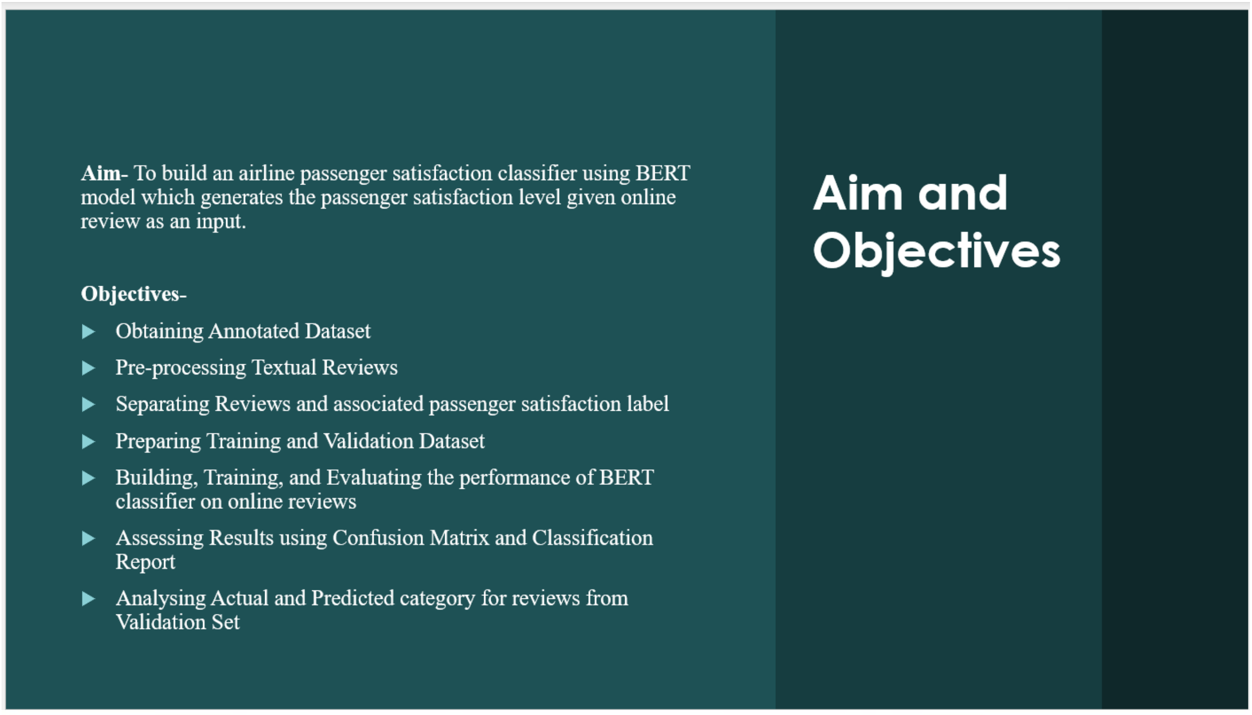


Research Questions

Are internet reviews more accurate and useful for predicting airline passenger pleasure than other information sources like surveys?

What are the most precise predicted terms or elements that affect the passenger experience in internet reviews?

Can a language-based machine learning model called BERT be trained to recognize problems that can suggest passenger annoyance, such as baggage handling, delays, etc.?

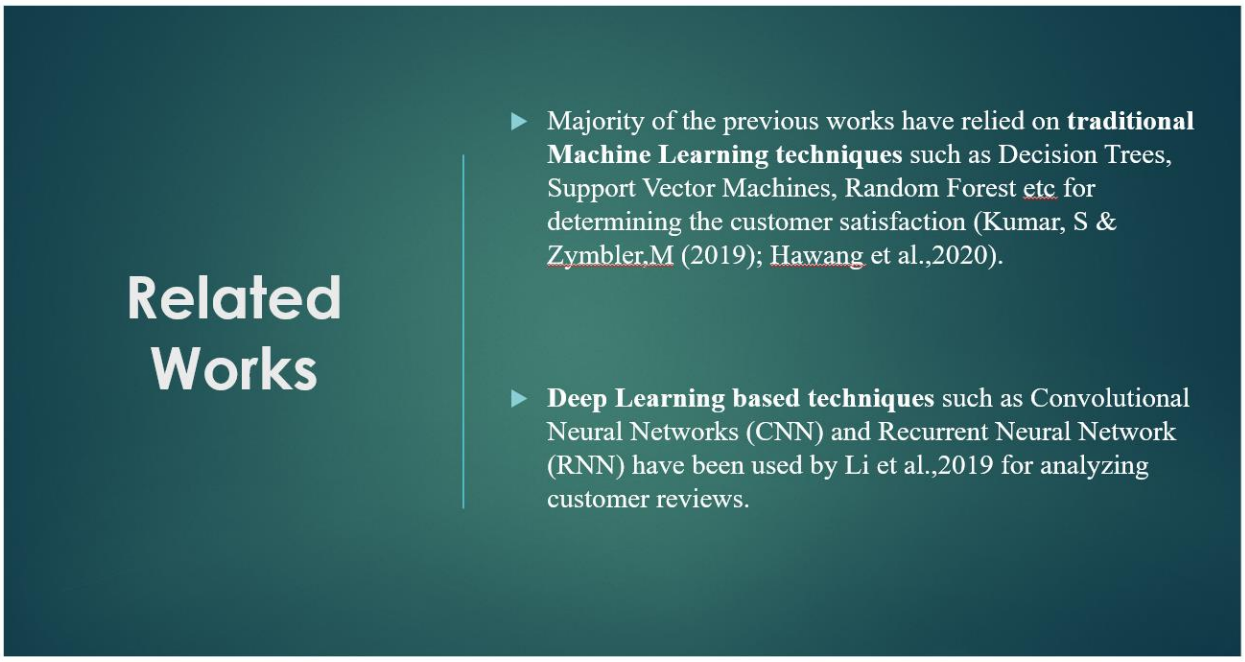


Aim and Objectives

Aim- To build an airline passenger satisfaction classifier using BERT model which generates the passenger satisfaction level given online review as an input.

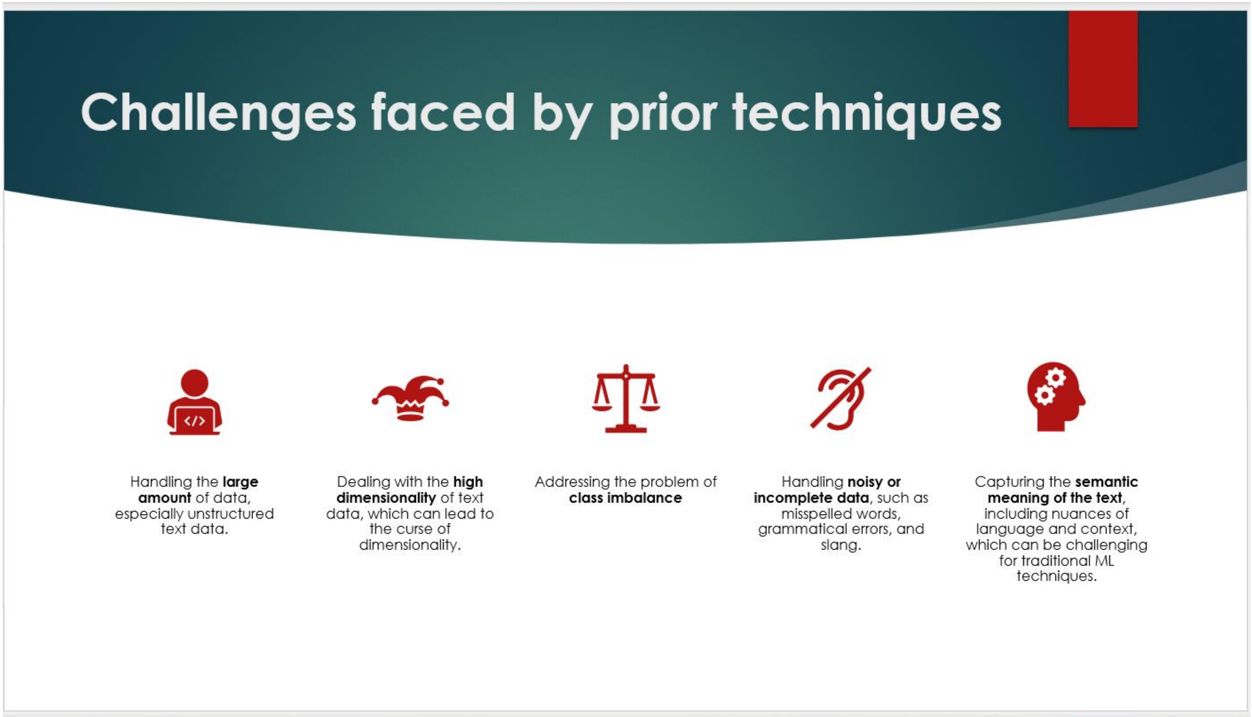
Objectives-

- Obtaining Annotated Dataset
- Pre-processing Textual Reviews
- Separating Reviews and associated passenger satisfaction label
- Preparing Training and Validation Dataset
- Building, Training, and Evaluating the performance of BERT classifier on online reviews
- Assessing Results using Confusion Matrix and Classification Report
- Analysing Actual and Predicted category for reviews from Validation Set



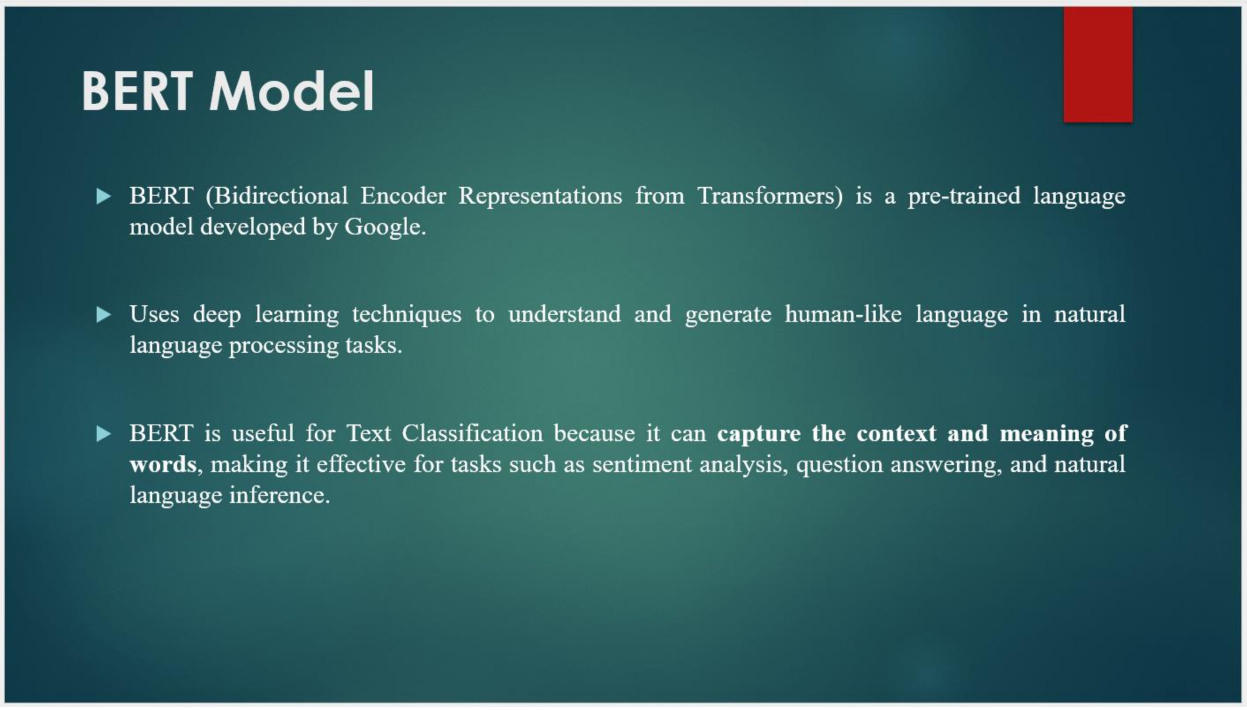
Related Works

- Majority of the previous works have relied on **traditional Machine Learning techniques** such as Decision Trees, Support Vector Machines, Random Forest etc for determining the customer satisfaction (Kumar, S & Zymbler,M (2019); Hawang et al.,2020).
- **Deep Learning based techniques** such as Convolutional Neural Networks (CNN) and Recurrent Neural Network (RNN) have been used by Li et al.,2019 for analyzing customer reviews.



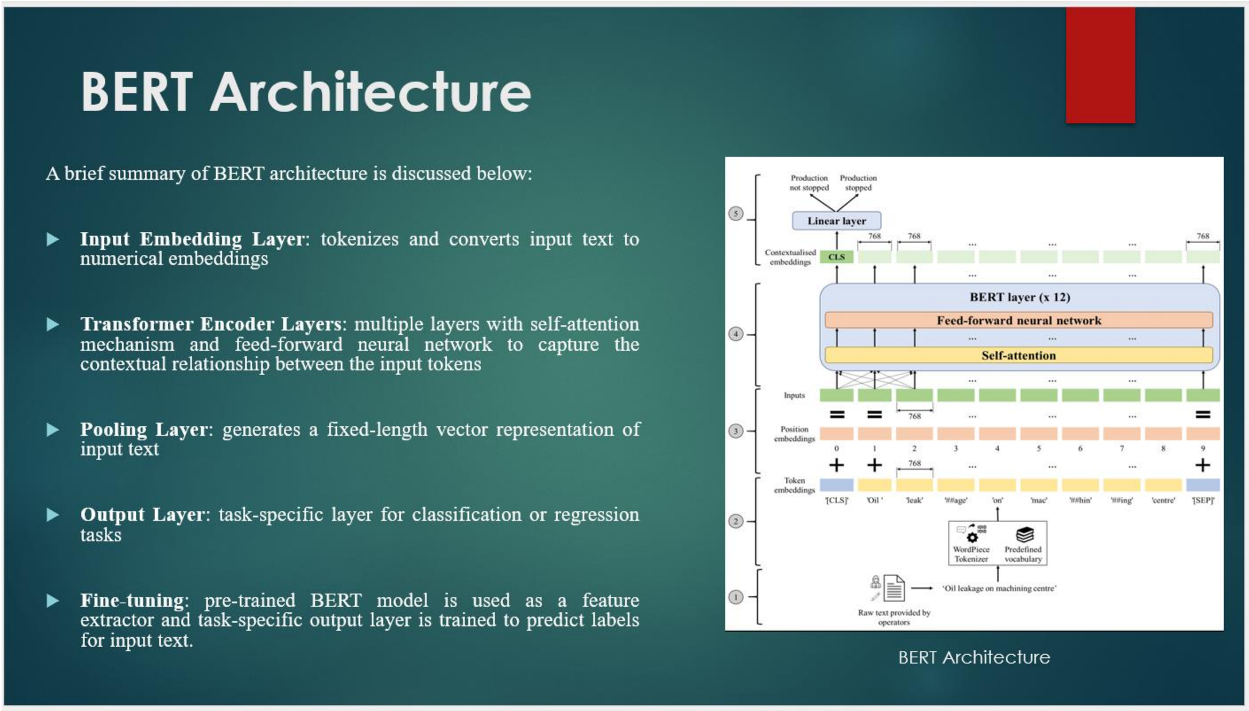
Challenges faced by prior techniques

- Handling the **large amount** of data, especially unstructured text data.
- Dealing with the **high dimensionality** of text data, which can lead to the curse of dimensionality.
- Addressing the problem of **class imbalance**
- Handling **noisy or incomplete data**, such as misspelled words, grammatical errors, and slang.
- Capturing the **semantic meaning of the text**, including nuances of language and context, which can be challenging for traditional ML techniques.



BERT Model

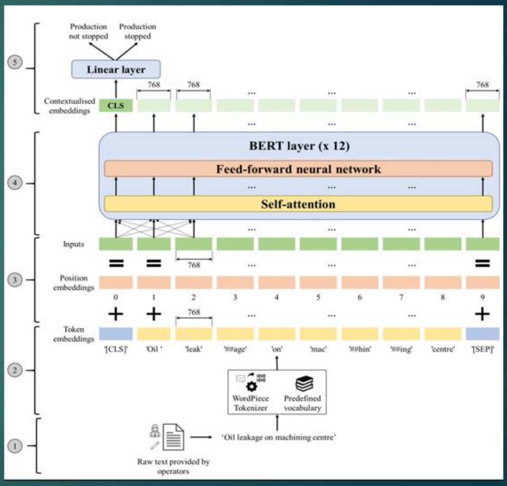
- BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model developed by Google.
- Uses deep learning techniques to understand and generate human-like language in natural language processing tasks.
- BERT is useful for Text Classification because it can **capture the context and meaning of words**, making it effective for tasks such as sentiment analysis, question answering, and natural language inference.



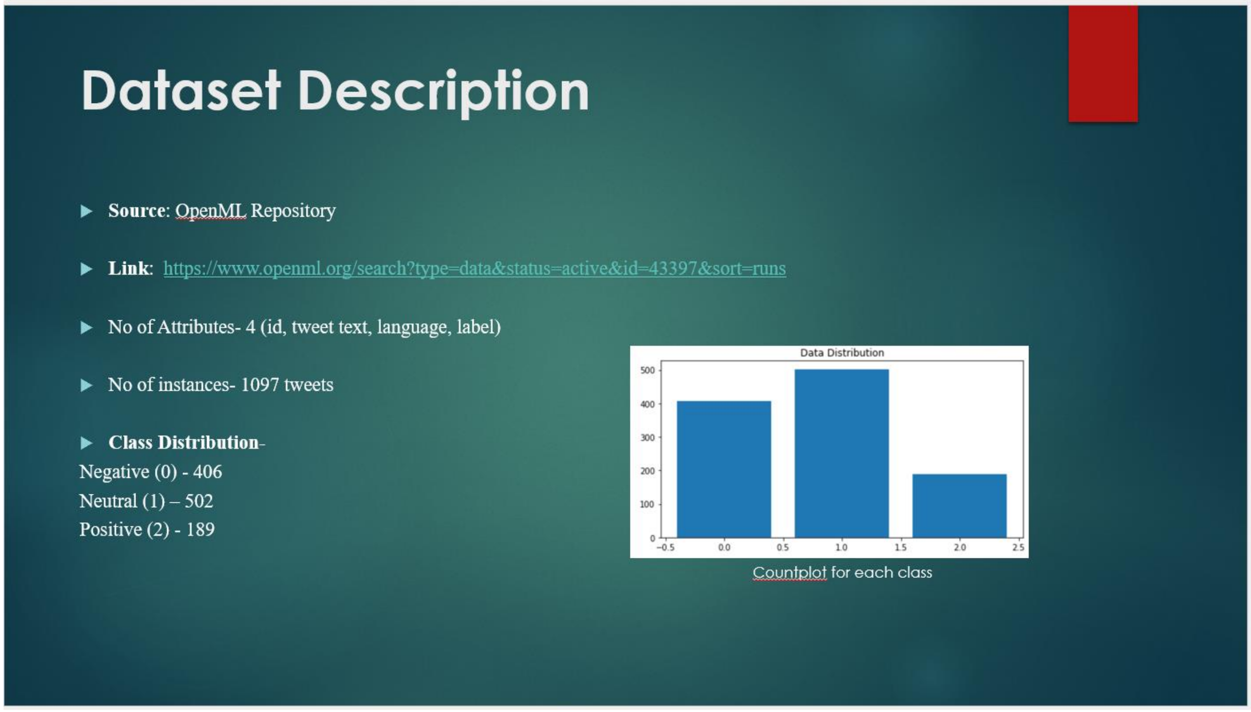
BERT Architecture

A brief summary of BERT architecture is discussed below:

- **Input Embedding Layer**: tokenizes and converts input text to numerical embeddings
- **Transformer Encoder Layers**: multiple layers with self-attention mechanism and feed-forward neural network to capture the contextual relationship between the input tokens
- **Pooling Layer**: generates a fixed-length vector representation of input text
- **Output Layer**: task-specific layer for classification or regression tasks
- **Fine-tuning**: pre-trained BERT model is used as a feature extractor and task-specific output layer is trained to predict labels for input text.



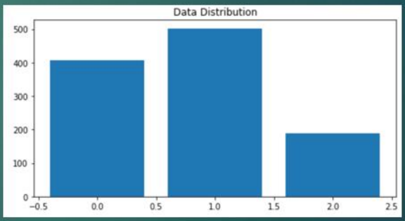
BERT Architecture



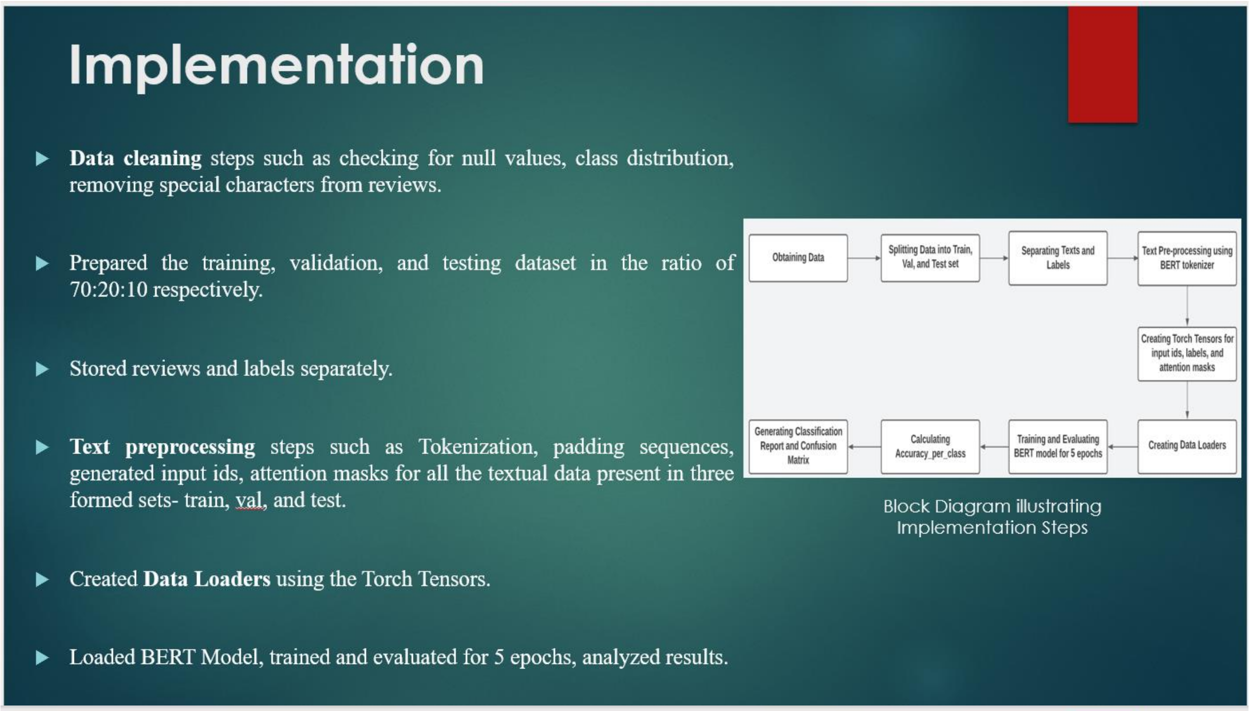
Dataset Description

- **Source**: OpenML Repository
- **Link**: https://www.openml.org/search?type=data&status=active&id=43397&sort=runs
- No of Attributes- 4 (id, tweet text, language, label)
- No of instances- 1097 tweets
- **Class Distribution**-

Negative (0) - 406
Neutral (1) – 502
Positive (2) - 189

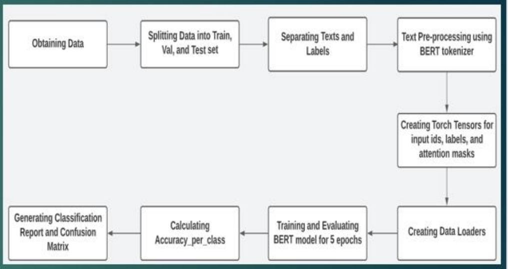


Countplot for each class

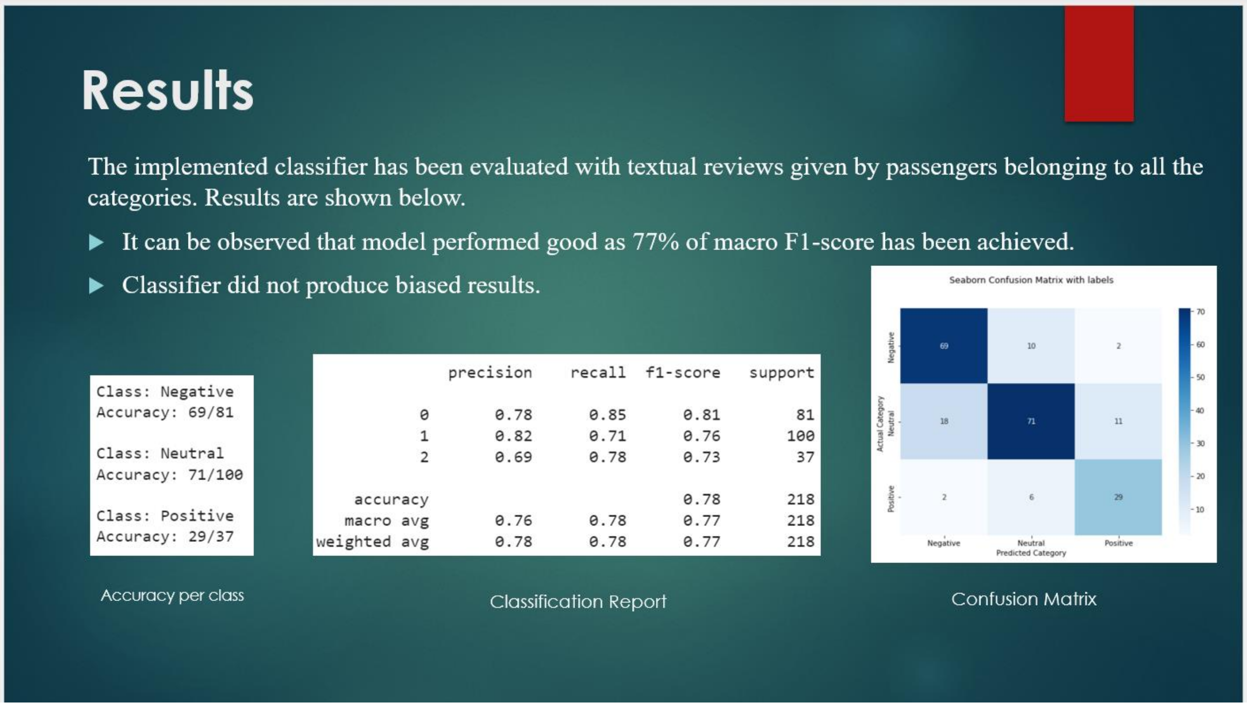


Implementation

- **Data cleaning** steps such as checking for null values, class distribution, removing special characters from reviews.
- Prepared the training, validation, and testing dataset in the ratio of 70:20:10 respectively.
- Stored reviews and labels separately.
- **Text preprocessing** steps such as Tokenization, padding sequences, generated input ids, attention masks for all the textual data present in three formed sets- train, val, and test.
- Created **Data Loaders** using the Torch Tensors.
- Loaded BERT Model, trained and evaluated for 5 epochs, analyzed results.


Obtaining Data
Splitting Data into Train, Val, and Test set
Separating Texts and Labels
Text Pre-processing using BERT tokenizer
Creating Torch Tensors for input ids, labels, and attention masks
Creating Data Loaders
Training and Evaluating BERT model for 5 epochs
Calculating Accuracy_per_class
Generating Classification Report and Confusion Matrix

Block Diagram illustrating Implementation Steps



Results

The implemented classifier has been evaluated with textual reviews given by passengers belonging to all the categories. Results are shown below.

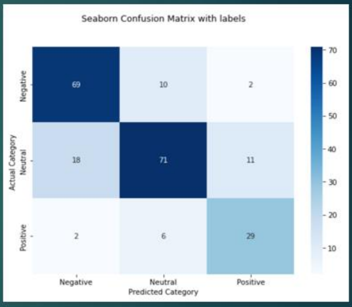
- It can be observed that model performed good as 77% of macro F1-score has been achieved.
- Classifier did not produce biased results.

```
Class: Negative
Accuracy: 69/81

Class: Neutral
Accuracy: 71/100

Class: Positive
Accuracy: 29/37
```

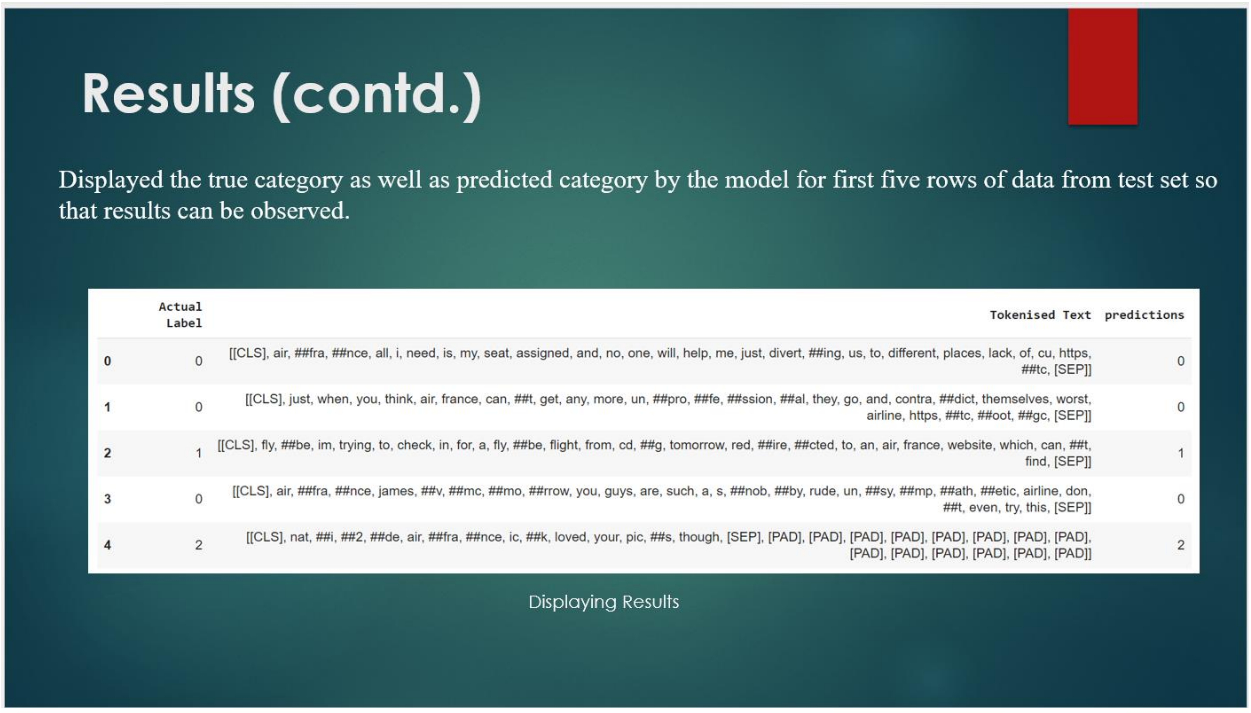
Accuracy per class

```
              precision    recall  f1-score   support

           0       0.78      0.85      0.81        81
           1       0.82      0.71      0.76       100
           2       0.69      0.78      0.73        37

    accuracy                           0.78       218
   macro avg       0.76      0.78      0.77       218
weighted avg       0.78      0.78      0.77       218
```

Classification Report



Confusion Matrix

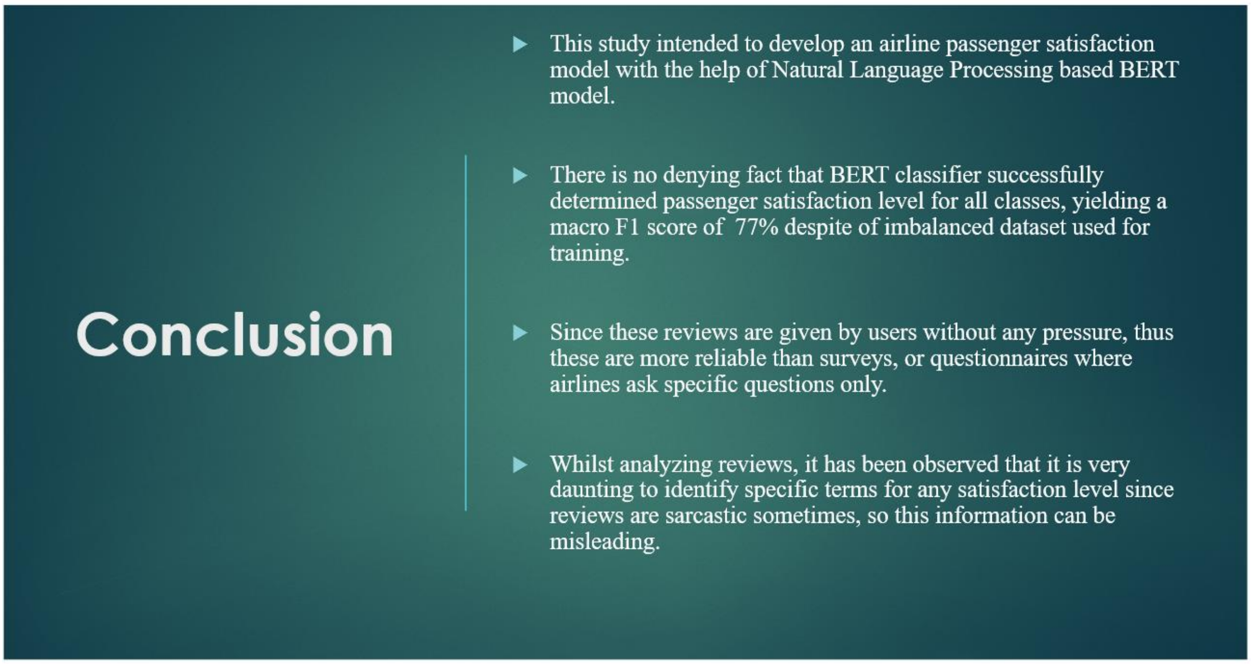


Results (contd.)

Displayed the true category as well as predicted category by the model for first five rows of data from test set so that results can be observed.

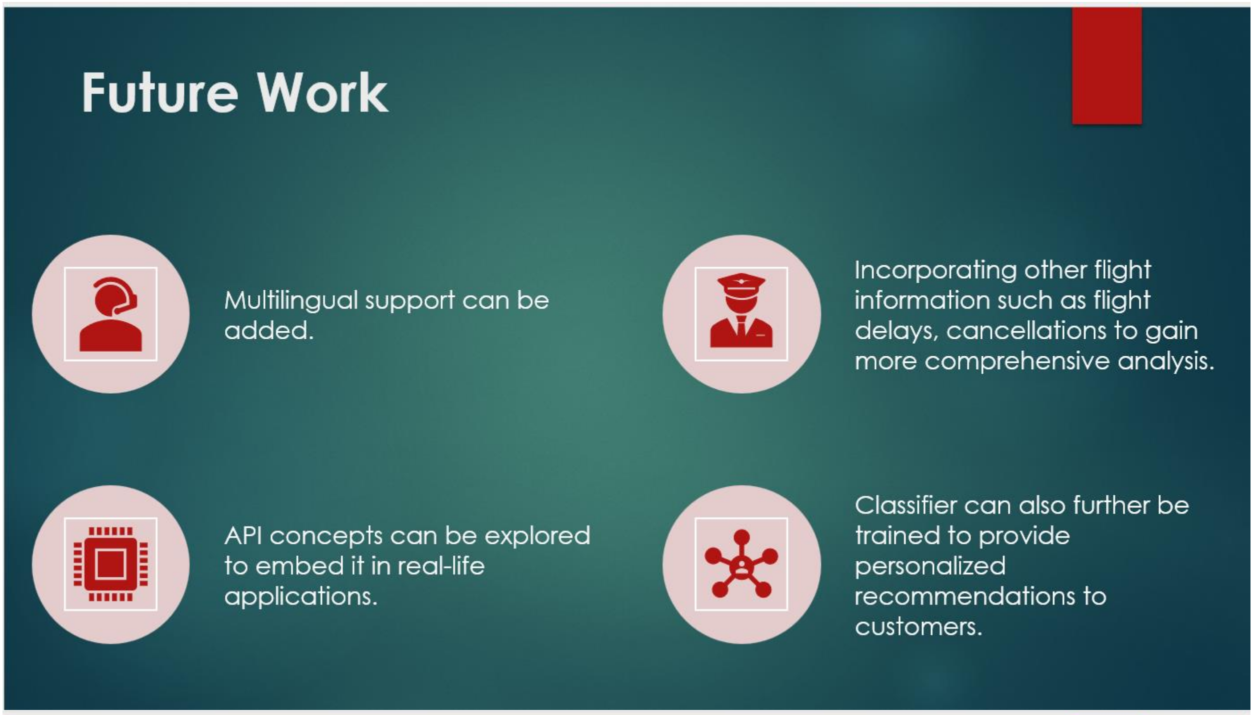
	Actual Label	Tokenised Text	predictions
0	0	[[CLS], air, ##fra, ##nce, all, i, need, is, my, seat, assigned, and, no, one, will, help, me, just, divert, ##ing, us, to, different, places, lack, of, cu, https, ##tc, [SEP]]	0
1	0	[[CLS], just, when, you, think, air, france, can, ##t, get, any, more, un, ##pro, ##fe, ##ssion, ##al, they, go, and, contra, ##dict, themselves, worst, airline, https, ##tc, ##oot, ##gc, [SEP]]	0
2	1	[[CLS], fly, ##be, im, trying, to, check, in, for, a, fly, ##be, flight, from, cd, ##g, tomorrow, red, ##ire, ##cted, to, an, air, france, website, which, can, ##t, find, [SEP]]	1
3	0	[[CLS], air, ##fra, ##nce, james, ##v, ##mc, ##mo, ##rrow, you, guys, are, such, a, s, ##nob, ##by, rude, un, ##sy, ##mp, ##ath, ##etic, airline, don, ##t, even, try, this, [SEP]]	0
4	2	[[CLS], nat, ##i, ##2, ##de, air, ##fra, ##nce, ic, ##k, loved, your, pic, ##s, though, [SEP], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD]]	2

Displaying Results



Conclusion

- This study intended to develop an airline passenger satisfaction model with the help of Natural Language Processing based BERT model.
- There is no denying fact that BERT classifier successfully determined passenger satisfaction level for all classes, yielding a macro F1 score of 77% despite of imbalanced dataset used for training.
- Since these reviews are given by users without any pressure, thus these are more reliable than surveys, or questionnaires where airlines ask specific questions only.
- Whilst analyzing reviews, it has been observed that it is very daunting to identify specific terms for any satisfaction level since reviews are sarcastic sometimes, so this information can be misleading.



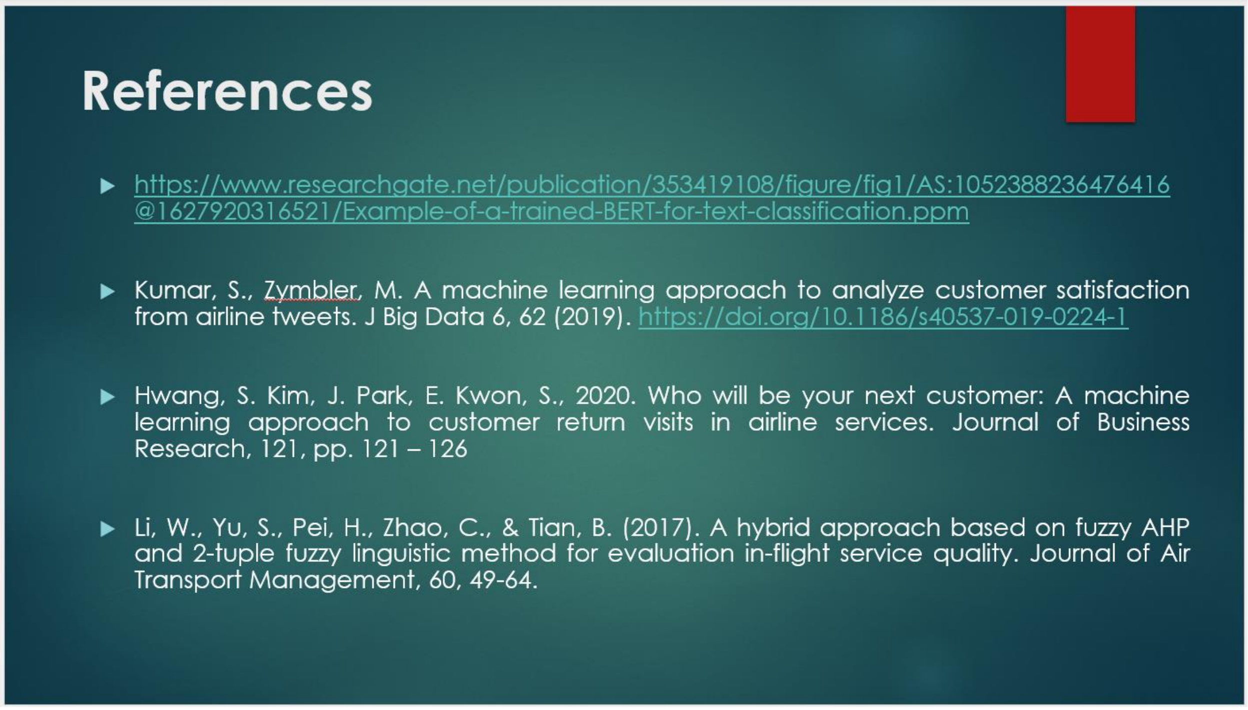
Future Work

Multilingual support can be added.

Incorporating other flight information such as flight delays, cancellations to gain more comprehensive analysis.

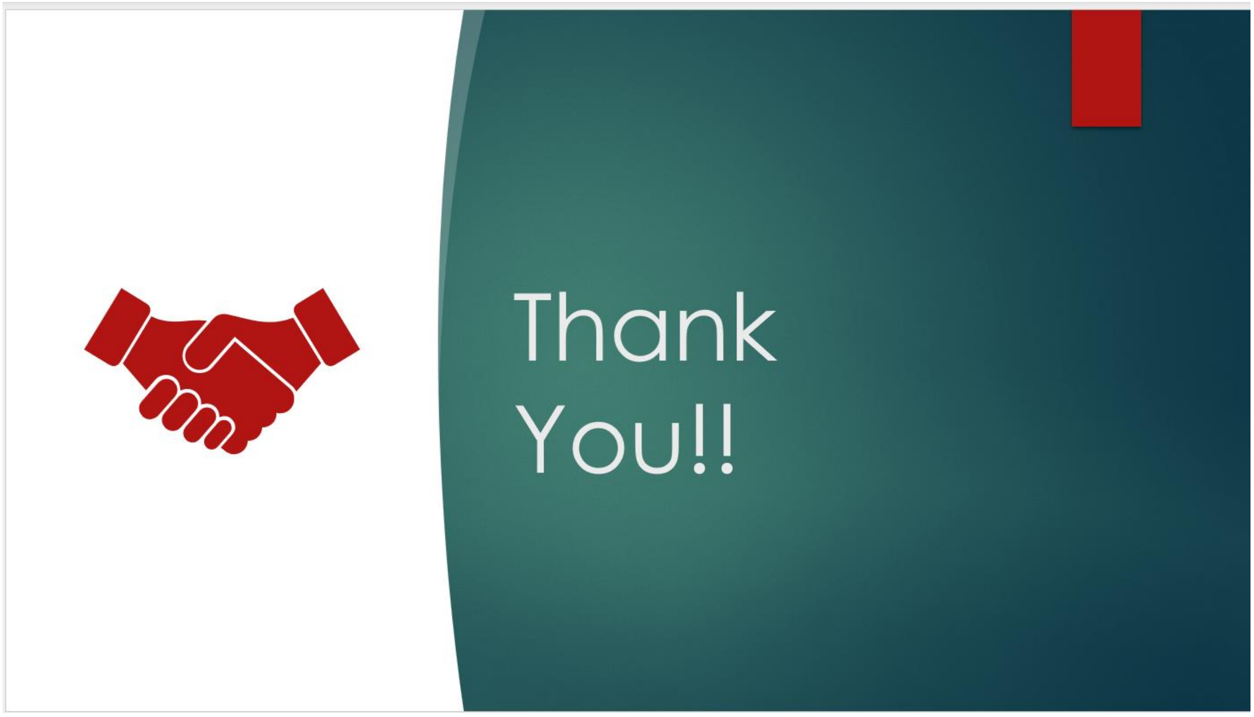
API concepts can be explored to embed it in real-life applications.

Classifier can also further be trained to provide personalized recommendations to customers.



References

- https://www.researchgate.net/publication/353419108/figure/fig1/AS:1052388236476416@1627920316521/Example-of-a-trained-BERT-for-text-classification.ppm
- Kumar, S., Zymbler, M. A machine learning approach to analyze customer satisfaction from airline tweets. J Big Data 6, 62 (2019). https://doi.org/10.1186/s40537-019-0224-1
- Hwang, S. Kim, J. Park, E. Kwon, S., 2020. Who will be your next customer: A machine learning approach to customer return visits in airline services. Journal of Business Research, 121, pp. 121 – 126
- Li, W., Yu, S., Pei, H., Zhao, C., & Tian, B. (2017). A hybrid approach based on fuzzy AHP and 2-tuple fuzzy linguistic method for evaluation in-flight service quality. Journal of Air Transport Management, 60, 49-64.



Thank You!!