Zagazig University

Faculty of Computers and Informatics

Artificial Intelligence Department

2023-2024

Project ID : GP_AI_2

Amazon product reviews Sentiment analysis

BY

1- Ahmed Shawaly
2- Ahmed Abdo
3- Eman Yousef
4- Sara Ahmed
5- Fatma Ahmed
6- Mohammad Amin

Under Supervision of

Dr. Gawaher Soliman

Eng. Walied Abdallah

ACKNOWLEDGMENT

We would like to express our deep gratitude to our supervisor, Dr. Gawaher Soliman, for giving us the opportunity to undertake this project under her supervision. Her guidance, support, and encouragement throughout the project have been invaluable. It was a great privilege and honor to work and study under her expert direction.

We would also like to extend our special thanks to Eng. Walied Abdullah, for his excellent support and significant contributions to the project. His insights, advice, and technical assistance have been crucial to the successful completion of this work. We are extremely grateful for the time, knowledge, and resources he has generously shared with us.

ABSTRACT

With the recent rapid spread of online shopping, we decided to create a website that make it easier for people to know if the product they want to buy is worth it or not, as a Manual analysis of reviews is time-consuming and impractical for large datasets. It is difficult to identify overall sentiment from numerous reviews. There is a need for a system to provide sentiment analysis and summarization of Amazon product reviews.

The aim of this project is to assist potential buyers who shop on Amazon.com by providing them with a clear and concise summary of product reviews. Our website just take the product link from the user then use Natural Language Processing (NLP) techniques to analyze the sentiment of the reviews associated with that product. By doing so, it delivers two key pieces of information: the percentage of positive and negative reviews and a summarized list of the pros and cons derived from these reviews to indicate the user whether the product has been liked by most users who have already tried the product or not, and a summarization of positive and negative reviews separately to make user know the pros and cons of the product.

Our system uses a sentiment analysis models, including BERT, DistilBERT, and RoBERTa, to ensure accurate and reliable sentiment classification. These models are fine-tuned on the Amazon Polarity Dataset to enhance their performance specifically for Amazon reviews. The analysis provides users with a clear indication of whether the product has been liked by the majority of users who have purchased and used it.

Additionally, the website presents a summarization of positive and negative reviews separately. This feature enables users to quickly understand the key advantages and disadvantages of the product, helping them to make a more informed decision. The visual representation of sentiment distribution further aids in providing a quick overview of the product's reception among users.

Our project aims to make online shopping more efficient and informed by providing a user-friendly tool that distills complex review data into actionable insights. By helping users quickly discern the overall sentiment and key points of feedback for a product, we enhance their ability to make confident purchasing decisions.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1:
INTRODUCTION

1.1 Overview

Now, we are in the age of speed where everyone wants to complete everything quickly. This is what our website “Analytica” does. We save user’s time, instead of reading a large number of reviews, trying to find out if the product is good to make him buy it and wasting time in asking about the product, we do it. Just put the product link in our website and take all of that easily.

We help you by providing some features that will meet your needs. You will be able to know positive and negative reviews percentage, which shows you whether the majority like the product or not. you also know the top repeated sentences and positive and negative reviews summarization, which will help you to know the advantages and disadvantages of the product you are confused about whether to buy or not.

The project is developed a web application to help online buyers to make a decision about buying a product. The way that project analyze the product reviews is using RoBERTa (language model based on BERT Transformer) and in summarizing use T5-small (Transformer).

1.2 Motivation

Online shopping statistics show the increase in digital order is not just a trend. We wanted to meet the users’ needs due to the increase of e-commerce market size around the world and the increase of online shopping. Choosing Amazon.com to start the project was not random, but because Amazon.com has the highest market share of all e-commerce companies. We help users by creating a website that provide some feature they want such as: percentage of positive and negative reviews, top repeated sentences and most important product advantages and disadvantages in the reviews summarization.

1.3 Main Problem

When shopping in Amazon, online buyers face difficulty in deciding whether to buy a product or not. They find that different types of one product have a high rate which makes the decision more difficult. When the user tried to know the advantages and disadvantages of the product, he find hundreds and thousands of reviews about the product. The buyers may is not able to read all the reviews about the product and reading some of these reviews will take a long time. They also can not manually analyzing large volumes of product reviews. This makes it difficult for potential buyers to make informed decision based on the collective reviews of previous customers. This make them need a tool to facilitate decision making and summarize all these reviews. That is exactly what we offer.

1.4 Solution

We have structured the project into two main sections to address the challenges associated with analyzing large volumes of product reviews on Amazon:

1. Reviews Classification:

To classify reviews as either positive or negative to provide users with a clear understanding of the general sentiment toward a product.We use :

•Deep Learning Models: Various deep learning models are employed to enhance the accuracy and efficiency of sentiment analysis.

•Natural Language Processing (NLP) Techniques: We leverage advanced NLP techniques for sentiment analysis to interpret the emotions and opinions expressed in the reviews.

•Sentiment Analysis: The sentiment analysis process involves preprocessing the text (such as tokenization, stop-word removal, and stemming) to prepare the data for analysis. Our models then evaluate each review and classify it as positive or negative.

•Percentage Calculation: Once all reviews for a given product are classified, we calculate the percentage of positive and negative reviews. This provides users with a quick, quantitative summary of the overall sentiment, making it easier for them to gauge the general reception of the product.

2. Reviews Summarization:

To summarize the most significant reviews to help users quickly understand the key strengths and weaknesses of a product.We use :

Transformer Models:

We utilize transformer models for the task of summarization, specifically:

•T5 (Text-To-Text Transfer Transformer) Small: T5 is used to generate summaries. It is particularly effective for text generation tasks due to its ability to understand and generate text in a coherent manner. We utilize the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score to evaluate the quality of the summaries generated by T5.

•BERT (Bidirectional Encoder Representations from Transformers): BERT is used in conjunction with the ROUGE score to further refine and improve the summarization process. BERT’s ability to understand context and semantics helps in generating more accurate and contextually relevant summaries.

By dividing our project into these two sections, we effectively address the challenges of manual review analysis. Our solution not only saves time and effort for consumers but also enhances their ability to make informed purchasing decisions. This systematic approach to sentiment analysis and review summarization leverages the power of NLP and deep learning to deliver meaningful insights from vast amounts of review data.

Chapter 2:
RELATED WORK

2.1 Overview

We will discuss the existing projects that try to achieve goals similar to our project “Analytica”. This section will clarify points of difference between our project and existing projects and highlights the innovations and improvements introduced by analytica.

2.2 Current Systems and issues

- **Current Systems :**

1- Arjun-Mota Amazon product reviews :

- Sentiment analysis of product reviews in positive and negative with identification of most reviewed products from Amazon product dataset.
- Do sentiment analysis on how many reviews are positive and negative based on rating given by the user (positive (>=3), otherwise is negative).
- Sentiment analysis is being done using the transformers.

System issues :

- It work in a static dataset and does not support scrapping.
- This system does not provide reviews summarization or user interface.

2- Watson-Second-Opinion :

- Create a Node.js app that takes the reviews from an online shopping website, Amazon, and feeds them into the Watson Natural Language Understanding service. The Watson Natural Language Understanding service will show the overall sentiments of the reviews. The sample application will do all the reading of reviews for you and will give an overall insight about them.
- Build a user interface around the result of Watson Natural Language Understanding.

System issues :

- This system does not provide reviews summarization.

3- AlaBay sentiment analysis amazon product reviews:

- NLP with NLTK for Sentiment analysis amazon Products Reviews.
- Use word2Vec embedding.

- Sentiment analysis classification is done by using LSTM.

System issues :

- This system does not provide reviews summarization.
- It does not support scraping, the user should copy review by review to classify one review at a time.

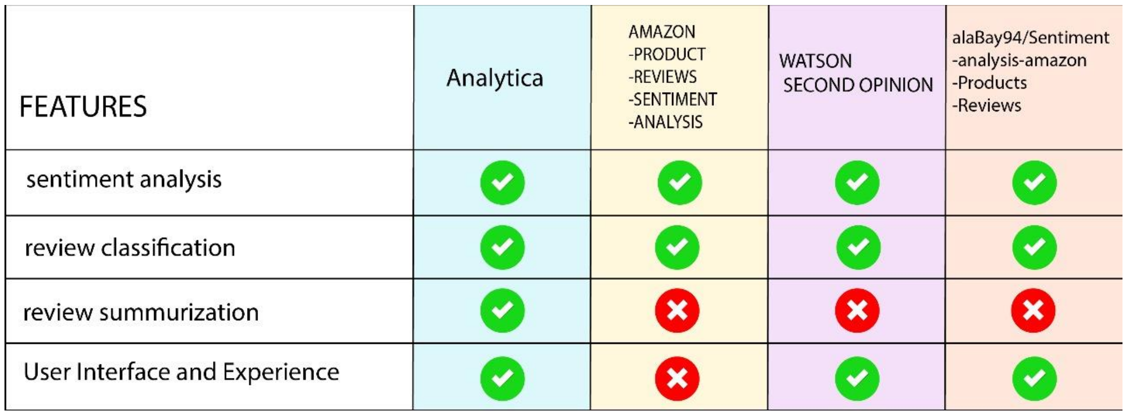
2.3 Our project features

Unlike the above systems, our project aims to solve these problems and provide integrated and more effective solutions. We added more features in our project to these systems. We provide :

- Efficient sentiment analysis to classify the scraped reviews into positive and negative reviews. We show that to the user in pie chart which displayed the percentage of positive and negative reviews to make it easier to define whether most of this product buyers liked it or not.
- We show the total number of reviews, number of positive reviews and number of negative reviews.
- Top repeated sentences on positive and negative reviews separately.
- Positive and negative reviews summarization separately.

2.4 Competitor Analysis

After doing our research about similar idea to our project. We found some projects that offer sentiment analysis of Amazon product reviews and classify them but no one added the summarization feature. We thought about what add-on the user would need for these projects and we found that by using reviews summarization we will save the user time.



FEATURES	Analytica	AMAZON -PRODUCT -REVIEWS -SENTIMENT -ANALYSIS	WATSON SECOND OPINION	alaBay94/Sentiment -analysis-amazon -Products -Reviews
sentiment analysis	✓	✓	✓	✓
review classification	✓	✓	✓	✓
review summurization	✓	✗	✗	✗
User Interface and Experience	✓	✗	✓	✓

Figure 2.1 : Competitor Analysis.

2.5 Other Papers

1- Analyzing Amazon Products Sentiment: A Comparative Study of Machine and Deep Learning, and Transformer-Based Techniques

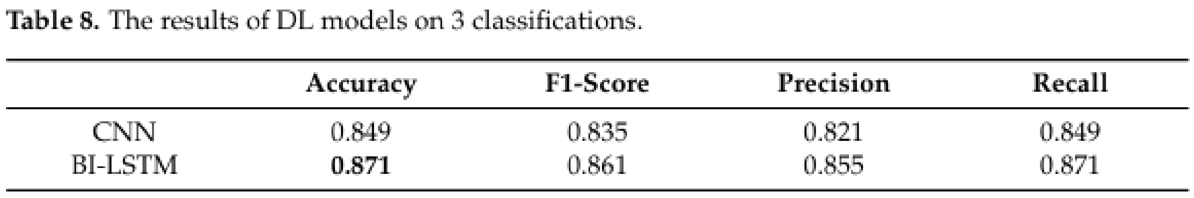
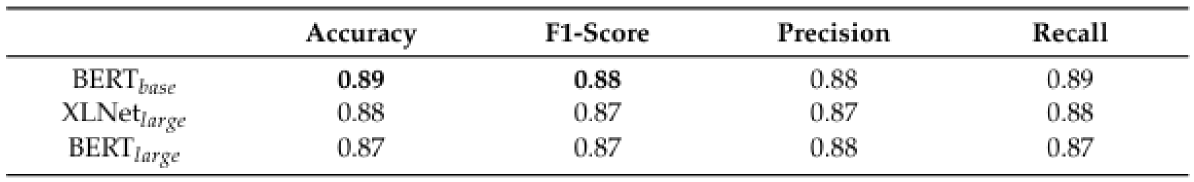


Table 8. The results of DL models on 3 classifications.

	Accuracy	F1-Score	Precision	Recall
CNN	0.849	0.835	0.821	0.849
BI-LSTM	**0.871**	0.861	0.855	0.871

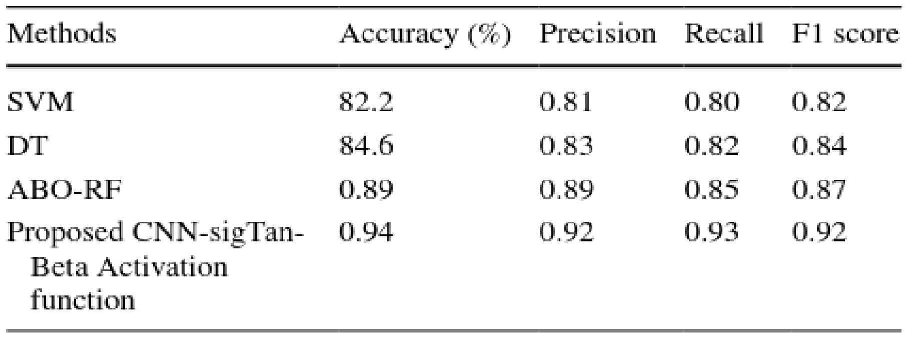
Figure 2.2 : The results of DL models "Paper 1".



	Accuracy	F1-Score	Precision	Recall
BERT$_{base}$	**0.89**	**0.88**	0.88	0.89
XLNet$_{large}$	0.88	0.87	0.87	0.88
BERT$_{large}$	0.87	0.87	0.88	0.87

Figure 2.3 : Paper 1 results.

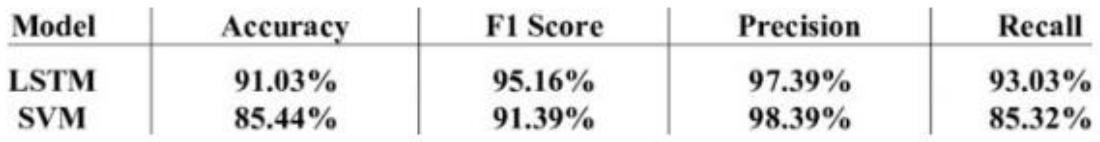
2- Enhancing sentiment analysis classification for amazon product reviews using CNN- sigTan-Beta activation function



Methods	Accuracy (%)	Precision	Recall	F1 score
SVM	82.2	0.81	0.80	0.82
DT	84.6	0.83	0.82	0.84
ABO-RF	0.89	0.89	0.85	0.87
Proposed CNN-sigTan-Beta Activation function	0.94	0.92	0.93	0.92

Figure 2.4 : Paper 2 results.

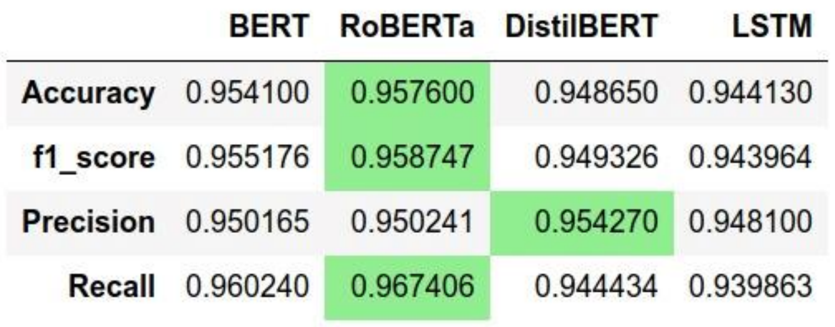
3- Sentiment analysis applied on Amazon reviews



Model	Accuracy	F1 Score	Precision	Recall
LSTM	91.03%	95.16%	97.39%	93.03%
SVM	85.44%	91.39%	98.39%	85.32%

Figure 2.5 : Paper 3 results.

4- Our Project “Analytica”



	BERT	RoBERTa	DistilBERT	LSTM
Accuracy	0.954100	0.957600	0.948650	0.944130
f1_score	0.955176	0.958747	0.949326	0.943964
Precision	0.950165	0.950241	0.954270	0.948100
Recall	0.960240	0.967406	0.944434	0.939863

Figure 2.6 : Our Project results.

Chapter 3:
SYSTEM ANALYSIS

3.1 Scenario

1) The user enter to home page and know our service.
2) User create a new account or login if he already has one.
3) User choose a product from Amazon.com and take its link.
4) User put the link in our website then click submit button.
5) We scrape reviews from product page.
6) The system analyze and summarize the reviews.
7) User read the results (positive and negative reviews percentage, top repeated sentences and reviews summarization) and decide whether he will buy it or not.

3.2 System requirements

Functional requirements :

- Login / Register.
- Input product link.
- Web scraping.
- Reviews analysis and summarization.
- Display the results.

Non-Functional requirements :

- **Operational :**
  - The website runs on web browser.

- **Performance :**
  - The system should take a few time in presenting the results.
  - The system should be available at any time.
  - The system should be accurate in results.

- **Security :**
  - Any user who makes use of the system need to hold a log in e-mail and password.

- **Culture and political :**
  - Personal information is protected in compliance with the Data Protection Act.

3.3 Methodologies

3.3.1 Algorithmic Components and Models :

1- **Sentiment Analysis Algorithms :**

- **Deep Learning Models :**
  - **Long Short-Term Memory Networks (LSTMs):**
    Specialized RNNs that mitigate the vanishing gradient problem.

- **Natural Language Processing Transformers :**
  - **BERT (Bidirectional Encoder Representations from Transformers :**
    Leverages a transformer-based neural network to understand and generate human-like language. BERT employs an encoder-only architecture. Fine-tuned on labeled data.
  - **RoBERTa (Robustly Optimized BERT Approach) :**
    Is a transformer-based language model that improves upon BERT. RoBERTa was trained on a much larger dataset and using a more effective training procedure.
  - **DistilBERT :**
    Is a small, fast, cheap and light transformer model trained by distilling BERT.

2- **Summarization Algorithms :**

- **Transformer Models :**
  - **T5 :**
    Is an encoder-decoder model pro-trained on a multi-task mixture of unsupervised and supervised tasks, and capture contextual information effectively.

  - **BERT :**
    A powerful pre-trained language model. Can be tuned to easily model that is naturally bidirectional, and capture contextual information.

3.3.2 Data Preprocessing Techniques :

1- **Text Cleaning and Normalization :**
  - **HTML Tag Removal :**
    Strip HTML tags from scraped data using libraries like BeautifulSoup.
  - **Lowercasing :**
    Convert all text to lowercase to ensure consistency.
  - **Tokenization :**
    Split text into tokens (words or subwords) for analysis.
  - **Stopword Removal :**
    Eliminate common words (e.g., 'and', 'the') that carry little sentiment information**.**
  - **Punctuation Removal :**
    Exclude punctuation marks as they typically do not contribute to sentiment analysis**.**

2- Handling Negations and Contractions :

- **Negation Handling :**
  Convert negated words (e.g., 'not good') into their respective antonyms ('not good' becomes 'not_bad').
- **Contractions :**
  Expand contractions (e.g., 'can't' becomes 'cannot') to ensure proper sentiment analysis.

3- Feature Extraction :

- **Bag-of-Words (BoW) :**
  Represent text as a collection of words without considering grammar or word
- **TF-IDF (Term Frequency-Inverse Document Frequency) :**
  Weigh words by their frequency in a document relative to their frequency across all documents.
- **Word Embeddings :**
  Represent words in vector space, capturing semantic relationships (e.g., Word2Vec, GloVe).

4- Handling Imbalanced Data :

- **Upsampling :**
  Increase the number of minority class samples.
- **Downsampling :**
  Decrease the number of majority class samples.
- **SMOTE (Synthetic Minority Over-sampling Technique :**
  Generate synthetic samples for the minority class.

3.4 Use case

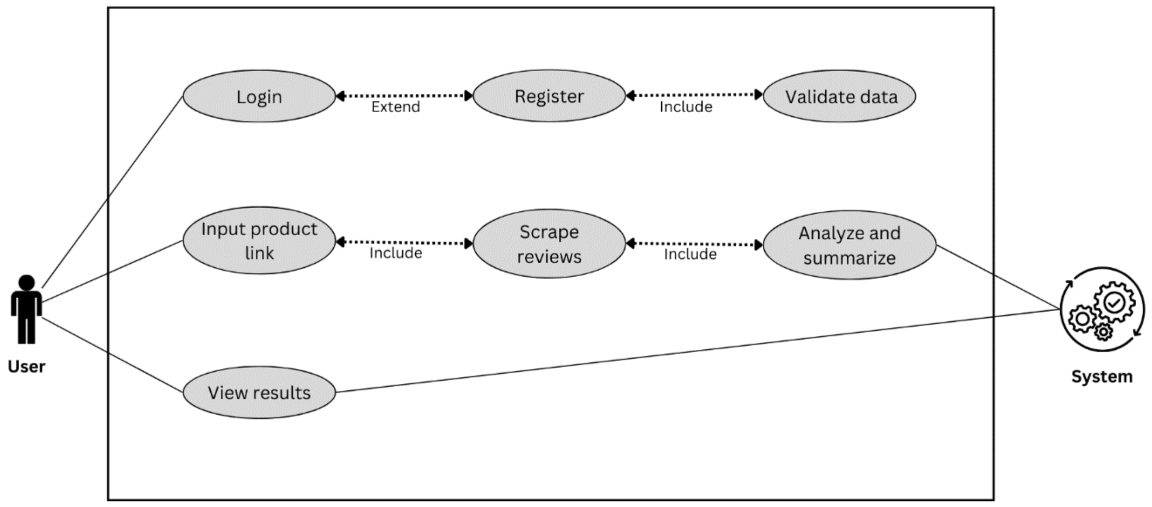


Figure 3.1 : Use Case.

Actors :

- User : the primary actor who interacts with the system. The user can register, login, input a product link and view the sentiment analysis result.
- System : represents the backend system that processes the user inputs, scrapes reviews, analyzes sentiments and return results.

Use cases :

- Register : the user registers on the platform by providing a username, email, and password.
- Validate data : validates the user input during the registration process to ensure all required fields are filled correctly.
- Login : the user logs in to the system using his email and password
- Input product link : the user inputs an Amazon product link to start the sentiment analysis process.
- Scrape reviews : the system scrapes reviews from provided Amazon product link.
- Analyze and summarize : the system analyzes the scraped reviews to determine the sentiment (positive or negative) and summarize the results.
- View results : the user views the results (percentage of positive and negative reviews, and reviews summarization).

3.5 Data flow

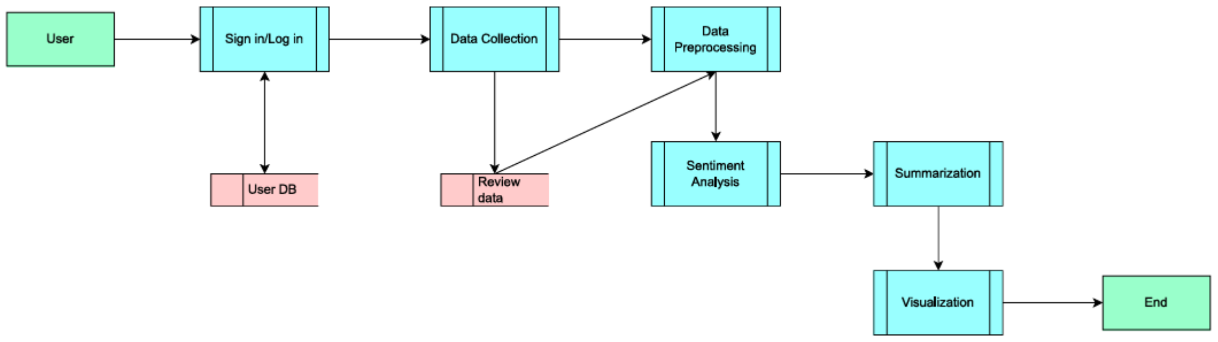
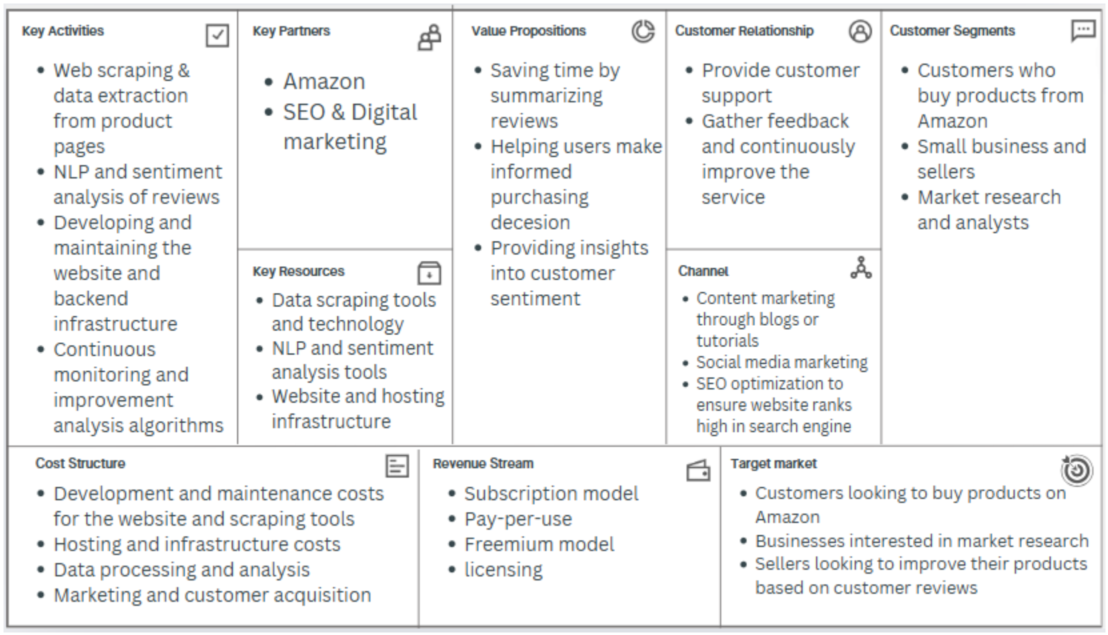


Figure 3.2 : Data Flow.

- Sign up/ Login : user provide credentials (username, email, and password). System provided credentials and authenticates the user. User data is stored and retrieved from the user database.
- Data collection : system retrieves product reviews from Amazon.com based on the product link provided by the user. Review data is stored in a review database.
- Data preprocessing : system performs text cleaning, normalization, and feature extraction on the collected reviews.

- Sentiment analysis : system processes the preprocessed reviews using deep learning models and NLP techniques to analyze sentiment. Review data is used as input for sentiment analysis.
- Summarization and visualization : system summarizes the reviews. Visualization component generates visual summaries of the sentiment analysis results.
- Output is the final visual summaries and sentiment analysis results.

3.6 Business Model



Key Activities
- Web scraping & data extraction from product pages
- NLP and sentiment analysis of reviews
- Developing and maintaining the website and backend infrastructure
- Continuous monitoring and improvement analysis algorithms

Key Partners
- Amazon
- SEO & Digital marketing

Key Resources
- Data scraping tools and technology
- NLP and sentiment analysis tools
- Website and hosting infrastructure

Value Propositions
- Saving time by summarizing reviews
- Helping users make informed purchasing decesion
- Providing insights into customer sentiment

Customer Relationship
- Provide customer support
- Gather feedback and continuously improve the service

Channel
- Content marketing through blogs or tutorials
- Social media marketing
- SEO optimization to ensure website ranks high in search engine

Customer Segments
- Customers who buy products from Amazon
- Small business and sellers
- Market research and analysts

Cost Structure
- Development and maintenance costs for the website and scraping tools
- Hosting and infrastructure costs
- Data processing and analysis
- Marketing and customer acquisition

Revenue Stream
- Subscription model
- Pay-per-use
- Freemium model
- licensing

Target market
- Customers looking to buy products on Amazon
- Businesses interested in market research
- Sellers looking to improve their products based on customer reviews

Figure 3.3 : Business Model.

Chapter 4: IMPLEMENTATION

4.1 System architecture

First, the user should sign up if he does not have an account.

This system is taking a product link from Amazon.com by user, then scrape the product reviews. The system analyze the reviews' sentiment and summarize them. Finally, view the results to the user.

4.1.1 System Architecture Diagram :

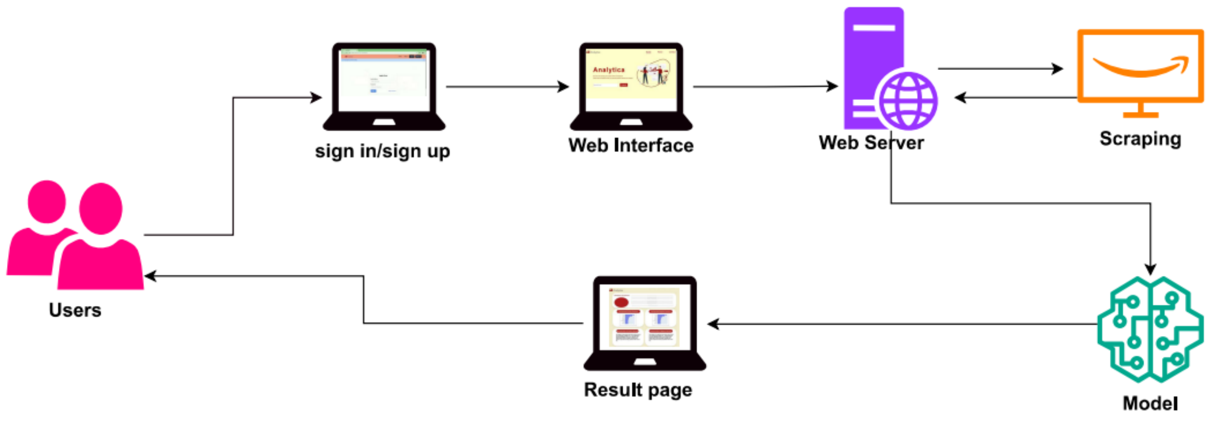


Figure 4.1 : System Architecture Diagram.

4.1.2 Technologies and Tools :

1- **Web Scraping :**
   - **BeautifulSoup & Scrapy :**
     For scraping product reviews from Amazon.com.

2- **Natural Language Processing (NLP) :**
   - **NLTK & Spacy** :
     For text preprocessing like tokenization, stop word removal, etc.

3- **Transformers :**
   - For advanced sentiment analysis and summarization using pre-trained models like T5 and BERT.

4- **Machine Learning :**
   - **Scikit-learn :**
     For building and evaluating machine learning models.
   - **TensorFlow & PyTorch :**
     For more advanced deep learning models.

5- **Data Visualization :**

- **Matplotlib & seaborn :**
  For plotting data and visualizing sentiment distribution.
- **Plotly :**
  For interactive visualization.

6- **Web Development :**

- **Flask :**
  For creating the web application backend.

7- **APIs :**

- **Amazon Product Advertising API :**
  For obtaining product details and reviews if scraping is not feasible.

4.2 System functionalities

4.2.1 Components:

- User Interface (UI) :
  Is the front-end component where users interact with the system. User interface primarily carries out two tasks :
  -Accepting user inputs.
  -Showing the results.
  User can input product link and see the analysis results (positive and negative percentage), top repeated sentences, and review summarization.
  We use technologies like : HTML, CSS and JavaScript.

  Responsibilities :

  -Provide a good and user-friendly interface for inputting product link.

  -Clear displaying of results.

- Web scraping :
  Is for extracting reviews from product link that user entered.
  Using : python, scrapy and ButifulSoup4.

- Sentiment analysis :
  Preprocesses the reviews text and processes and analyze sentiment in scraped reviews using a fine-tuned RoBERTa model to classify sentiment.
  Using : python, hugging face transformers.
- Review summarization module :
  Split the reviews into positive and negative. Summarize the reviews Using T5 (hugging face transformers).
- Backend system :

The backend system is the server side of the website, which connects processing tasks, including scraping, sentiment analysis, classification and summarization. It receives and handles user requests and communicates with the database and manage data flow. Using : flask,

4.2.2 Additional Features

1- Email Verification :
- Ensure account security by sending a verification link to the email address. Users cannot log in until their email is verified.

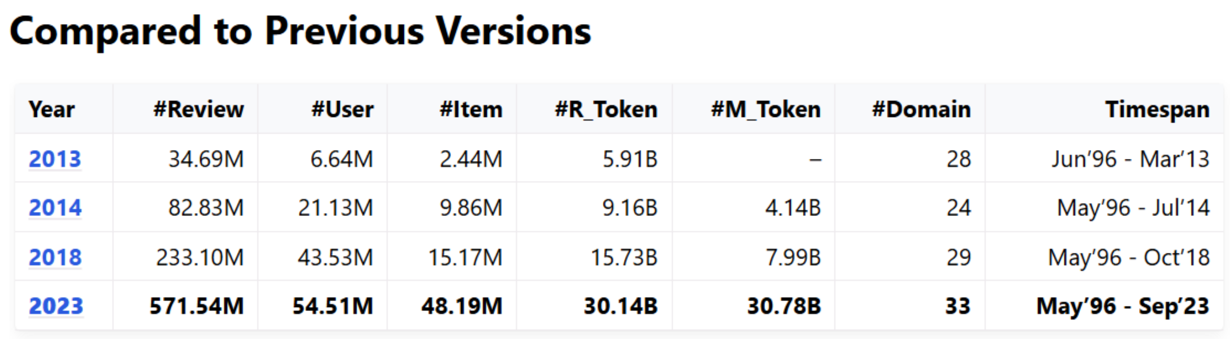
2- Secure Login :
- Users can only login after their account has been successfully verified.

3- User-Friendly interface :
- An intuitive user interface for signing up, logging in and verifying accounts.

4.3 Dataset

"Garbage in, garbage out." Keeping this adage in mind, we set out to find a suitable dataset for our model. Initially, we started with the Amazon product dataset, but we found that the data was highly imbalanced. We addressed this issue using the SMOTE technique and achieved an accuracy of 98% with a Random Forest model. However, the model did not perform well in production due to the limitations in vocabulary based on the Amazon product dataset.

Next, we moved to another dataset collected in 2023 by the McAuley Lab, which contains 571.54 million reviews from 54.51 million users and 48.19 million items, covering all categories on Amazon. Given the massive size of this dataset, we decided to focus on the electronics category, which contains 43.9 million reviews. This data was also imbalanced, so we randomly sampled 1.4 million reviews, ensuring a balanced distribution of ratings.



Compared to Previous Versions

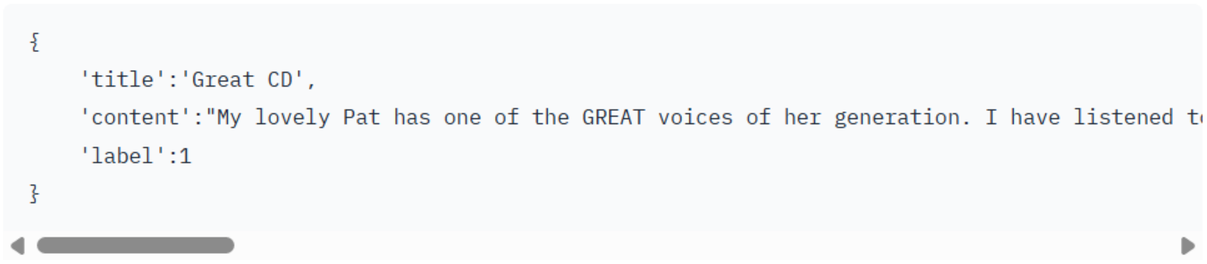
Year	#Review	#User	#Item	#R_Token	#M_Token	#Domain	Timespan
2013	34.69M	6.64M	2.44M	5.91B	–	28	Jun'96 - Mar'13
2014	82.83M	21.13M	9.86M	9.16B	4.14B	24	May'96 - Jul'14
2018	233.10M	43.53M	15.17M	15.73B	7.99B	29	May'96 - Oct'18
2023	**571.54M**	**54.51M**	**48.19M**	**30.14B**	**30.78B**	**33**	**May'96 - Sep'23**

Figure 4.2 : Dataset versions.

Supported Tasks and Leaderboards :

Text-classification, sentiment-classification: the dataset is mainly used for text classification: given the content and the title, predict the correct star rating.

Data instances : A typical data point, comprises of a title, a content and the corresponding label.



```
{
    'title':'Great CD',
    'content':"My lovely Pat has one of the GREAT voices of her generation. I have listened t
    'label':1
}
```

Figure 4.3 : Example of the dataset.

Data Fields :

'title': a string containing the title of the review - escaped using double quotes (") and any internal double quote is escaped by 2 double quotes (""). New lines are escaped by a backslash followed with an "n" character, that is "\n".

'content': a string containing the body of the document - escaped using double quotes (") and any internal double quote is escaped by 2 double quotes (""). New lines are escaped by a backslash followed with an "n" character, that is "\n".

'label': either 1 (positive) or 0 (negative) rating.

Despite this, we faced another issue: the distribution of review lengths was unclear. We split the dataset into two parts based on the mean sequence length and created a model for each part. The model for sequences shorter than the mean achieved an accuracy of 95%, while the model for longer sequences achieved an accuracy of 92%.

We then discovered the Amazon Polarity dataset, constructed by Xiang Zhang (xiang.zhang@nyu.edu). This dataset contains 4 million reviews, randomly sampled from the McAuley dataset. What distinguishes this data is the clear distribution of sequence lengths and balanced labels across various categories. We decided to take 1 million reviews from it using a random sampling technique for our model.

The Amazon reviews polarity dataset is constructed by taking review score 1 and 2 as negative, and 4 and 5 as positive. Samples of score 3 is ignored. Each class has 3,600,000 training samples and 400,000 testing samples. This data is balanced. It contain exactly 50% positive reviews and 50% negative reviews.

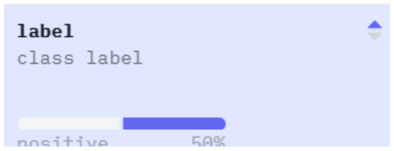


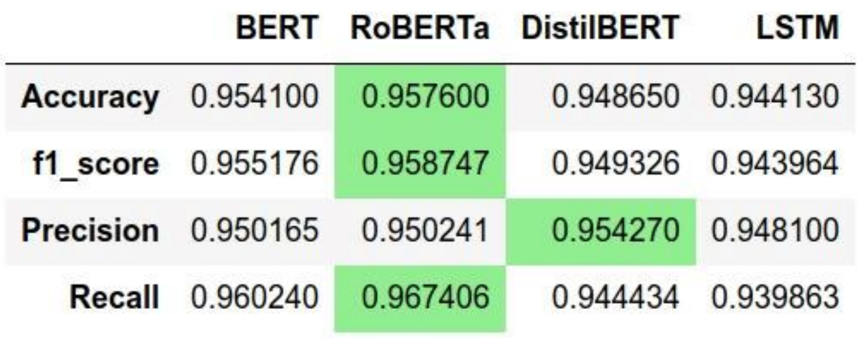
Figure 4.4 : Dataset classes.

4.4 Deep learning Models

4.4.1 Sentiment analysis :

- **Problem definition :**
  In our case, the problem we will be exploring is Binary Classification. This is because we are going to analyze product reviews to classify it to positive and negative reviews.
  For LSTM the train dataset contain 800,000 review, test dataset contain 100,000 reviews and validation dataset contain 100,000.
  And for pre-trained models the train is 80,000 and the validation is 20,000.

- **Algorithms comparison :**

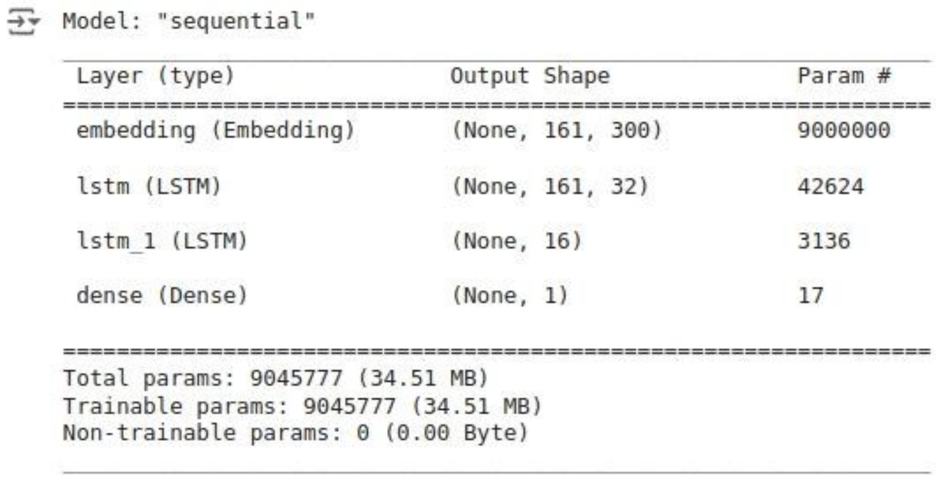


	BERT	RoBERTa	DistilBERT	LSTM
Accuracy	0.954100	0.957600	0.948650	0.944130
f1_score	0.955176	0.958747	0.949326	0.943964
Precision	0.950165	0.950241	0.954270	0.948100
Recall	0.960240	0.967406	0.944434	0.939863

Figure 4.5 : Algorithms Comparison.

1- Long Short-Term Memory Networks (LSTM) :

- Layers :



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 161, 300)          9000000

 lstm (LSTM)                 (None, 161, 32)           42624

 lstm_1 (LSTM)               (None, 16)                3136

 dense (Dense)               (None, 1)                 17

=================================================================
Total params: 9045777 (34.51 MB)
Trainable params: 9045777 (34.51 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 4.6 : LSTM summary (layers).

- Hyperparameters :

- For compile the model :
  optimizer = adam
  loss = binary_crossentroby
  metrics = accuracy
- For train the model :
  batch _size = 128
  epochs = 10
  validation_split = 0.2

- Confusion Matrix :

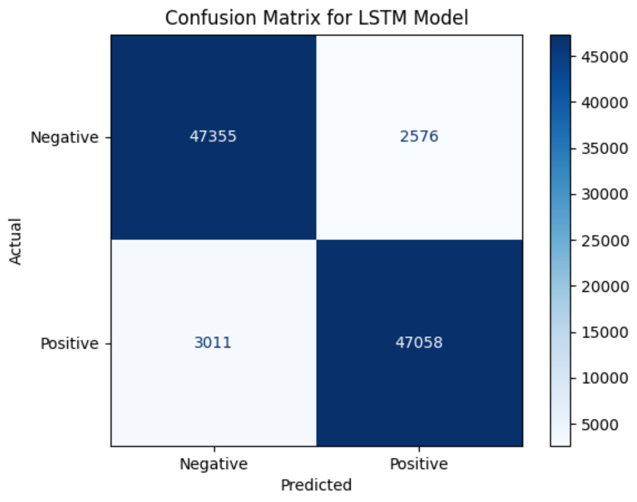


Figure 4.7 : LSTM Confusion Matrix.

2- Bidirectional Encoder Representation from Transformers (BERT):
Is a pre-trained model that can be fine-tuned for various natural language processing tasks. stands as an open-source machine learning framework designed for the realm of natural language processing (NLP).

- Hyperparameter :

Learning rate = 2e-5
Batch size = 16
Epochs = 1
Max sequence length = 200
Dropout weight = 0.1
Weight decay = 0.01

- Confusion Matrix :

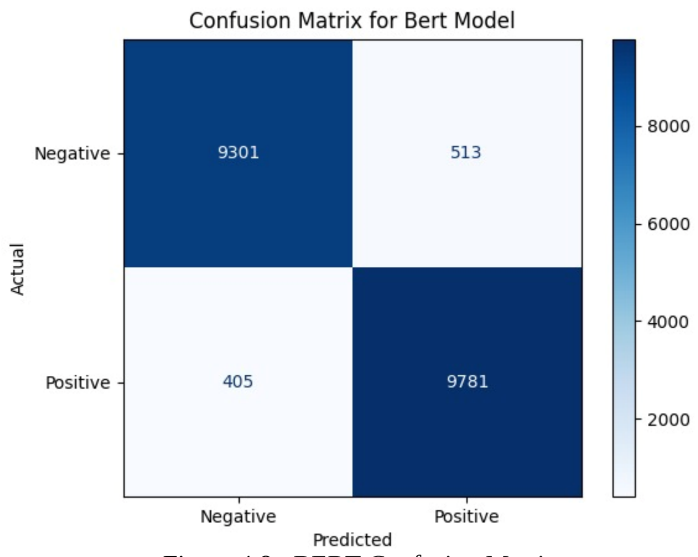


Figure 4.8 : BERT Confusion Matrix.

3- Robustly optimized BERT approach (RoBERTa) :

Is a language model based on BERT, but with different hyperparameters and training scheme. One key difference between RoBERTa and BERT is that RoBERTa was trained on a much larger dataset and using a more effective training procedure.

- Hyperparameter :

Learning rate = 2e-5

Batch size = 16

Epochs = 4

Max sequence length = 200

Dropout weight = 0.1

Weight decay = 0.01

- Confusion Matrix :

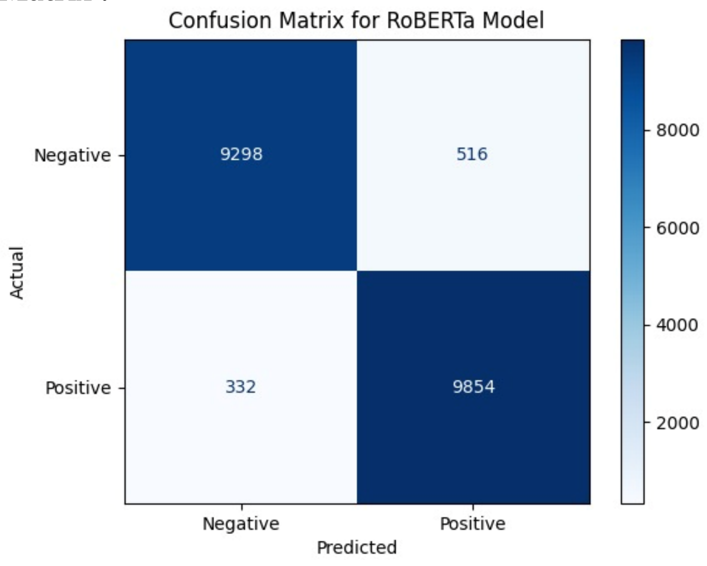


Figure 4.9 : RoBERTa Confusion Matrix.

4- DistilBERT :

Is a small, fast and light Transformer model trained by distilling BERT base. Same as BERT but smaller. Trained by distillation of the pretrained BERT model, meaning it's been trained to predict the same probabilities as the larger model.

- Hyperparameter :

Learning rate = 2e-5
Batch size = 16
Epochs = 4
Max sequence length = 200
Dropout weight = 0.1
Weight decay = 0.01

- Confusion Matrix :

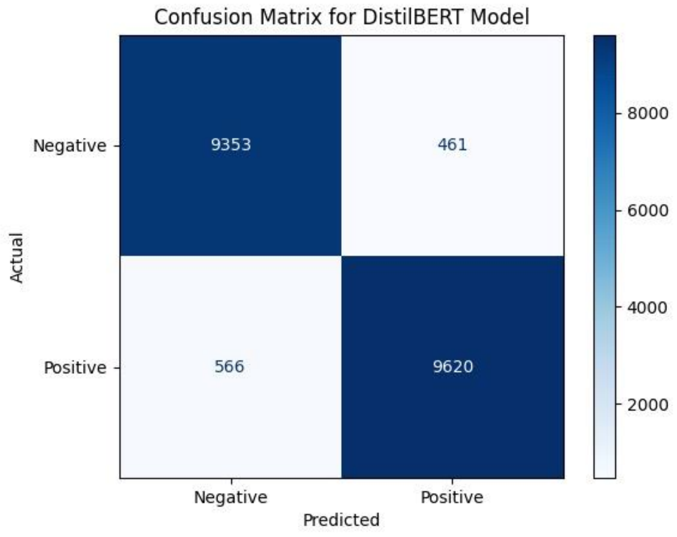


Figure 4.10 : DistilBERT Confusion Matrix.

4.3.2 Summarization:

1- Problem definition :

In this case, the problem is summarize the product reviews to provide a perception to the user about it and save his time instead of reading a large number of reviews .
The dataset contain 600017 reviews, consisting of (30010 positive and 300007 negative).

2- Algorithms comparison :

We use Recall-Oriented Understudy for Gisting Evaluation (ROUGE) ,is a set of metrics used for evaluating automatic summarization and machine translation software in natural language processing.

ROUGE-1

“considering 1-grams only”.

	T5-small	**BERT**
Recall	0.72	0.039
Precision	0.667	0.933
F1-score	0.692	0.076

Table 4.1 : 1-grams.

ROUGE-2

“considering 2-grams only”.

	T5-small	**BERT**
Recall	0.627	0.013
Precision	0.521	0.722
F1-score	0.569	0.025

Table 4.2 : 2-grams.

ROUGE-L

“is based on the longest common subsequence (LCS)”.

	T5-small	**BERT**
Recall	0.72	0.039
Precision	0.667	0.916
F1-score	0.692	0.075

Table 4.3 : longest common subsequence (LCS).

T5-small :

Hyperparameter :

1- **Device selection :**
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

2- **Tokenization :**
max_length=400

3- **Generated summary :**
max_length=150
min_length=40
length_penalty=2.0
num_beams=4

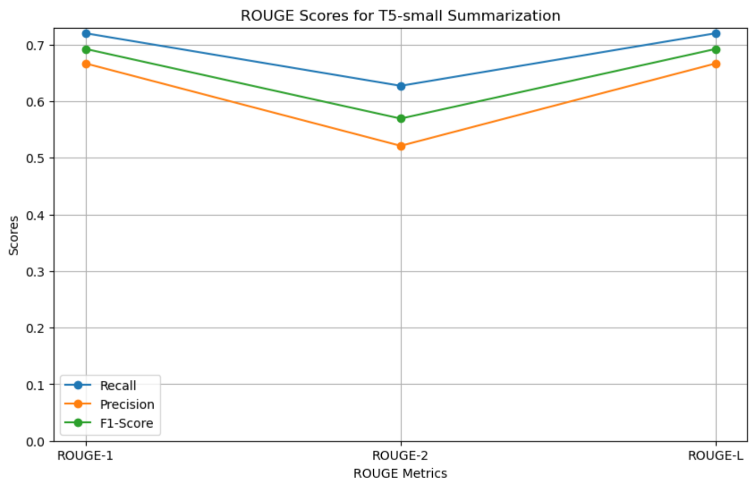


Figure 4.11 : Rouge Scores for T5-small.

BERT :

Hyperparameters :

1- Summarization ration of the original text :
ratio=0.2
2- Number of sentences :
num_sentences=3

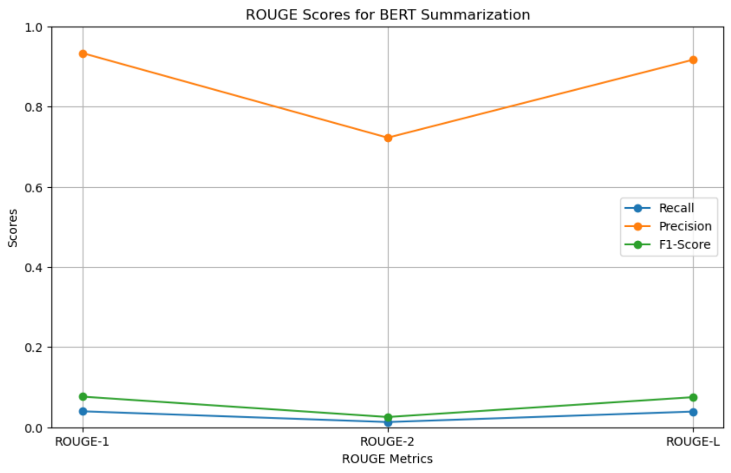
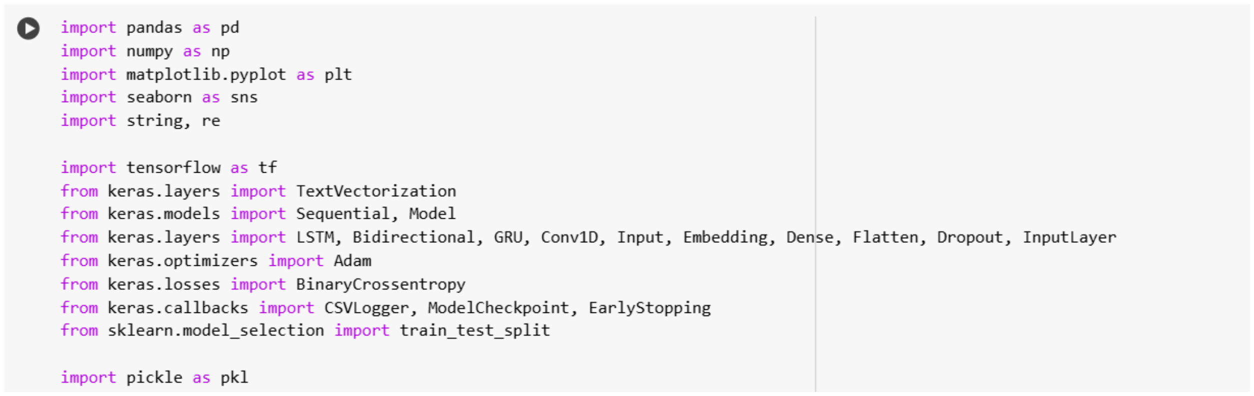


Figure 4.12 : Rouge scores for BERT.

4.5 Implementation and Testing

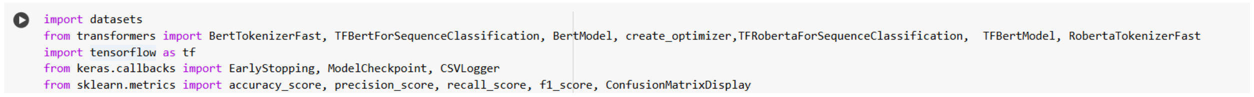
4.5.1 Sentiment Analysis Models

4.5.1.1 Preparing the Tools



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string, re

import tensorflow as tf
from keras.layers import TextVectorization
from keras.models import Sequential, Model
from keras.layers import LSTM, Bidirectional, GRU, Conv1D, Input, Embedding, Dense, Flatten, Dropout, InputLayer
from keras.optimizers import Adam
from keras.losses import BinaryCrossentropy
from keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping
from sklearn.model_selection import train_test_split

import pickle as pkl
```

Figure 4.13 : “import libraries” Code.



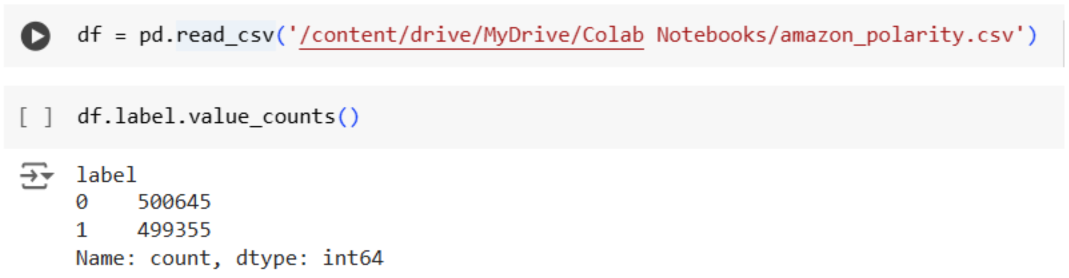
```
import datasets
from transformers import BertTokenizerFast, TFBertForSequenceClassification, BertModel, create_optimizer,TFRobertaForSequenceClassification,  TFBertModel, RobertaTokenizerFast
import tensorflow as tf
from keras.callbacks import EarlyStopping, ModelCheckpoint, CSVLogger
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, ConfusionMatrixDisplay
```

Figure 4.14 : Cont. “import libraries” Code.

4.5.1.2 Preparing the Data

1- Read the data :

We use Amazon polarity dataset from hugging face.



```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/amazon_polarity.csv')
```

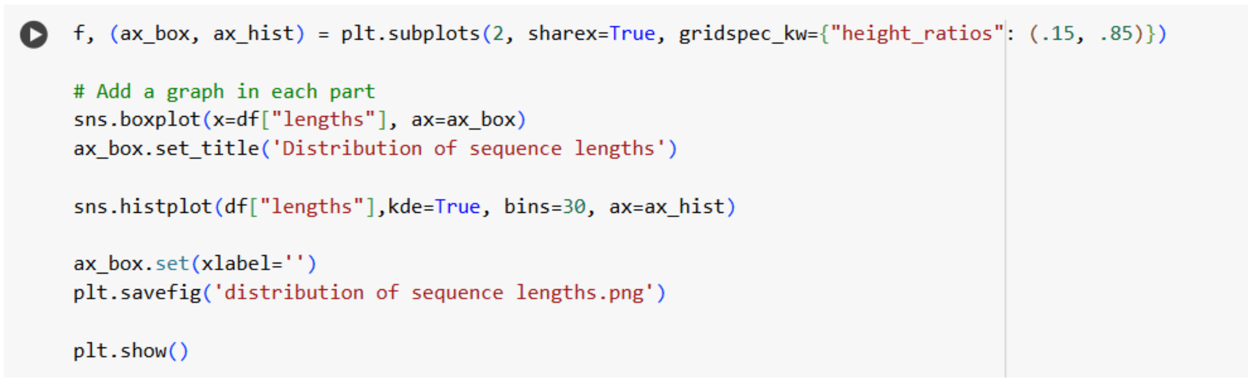
```
[ ] df.label.value_counts()
```

```
label
0    500645
1    499355
Name: count, dtype: int64
```

Figure 4.15 : “reading data” Code.

2- Data Visualization :

Here, we visualize distribution of reviews sequence length to understanding the range and common lengths of sequences in our dataset, which is essential for setting an appropriate sequence length (SEQUENCE_LEN) for text processing tasks.



```
f, (ax_box, ax_hist) = plt.subplots(2, sharex=True, gridspec_kw={"height_ratios": (.15, .85)})

# Add a graph in each part
sns.boxplot(x=df["lengths"], ax=ax_box)
ax_box.set_title('Distribution of sequence lengths')

sns.histplot(df["lengths"],kde=True, bins=30, ax=ax_hist)

ax_box.set(xlabel='')
plt.savefig('distribution of sequence lengths.png')

plt.show()
```

Figure 4.16 : “distribution of sequence length” Code.

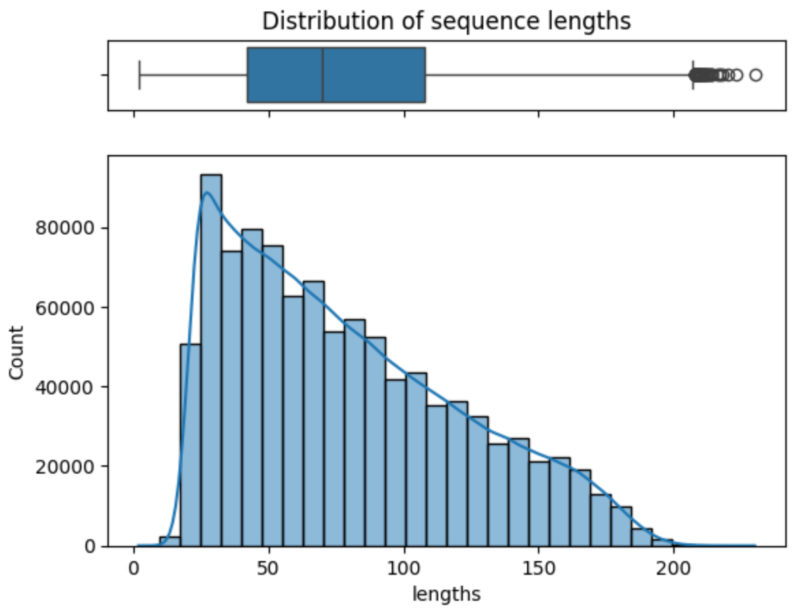
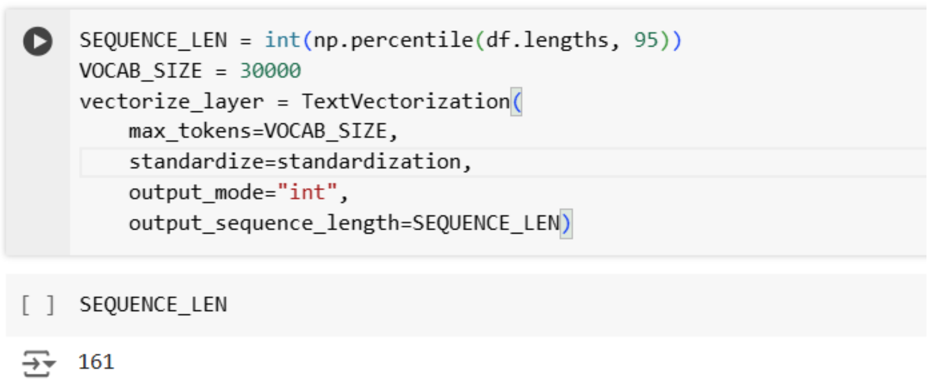


Figure 4.17 : Distribution of sequence length.

We calculate the 95th percentile of the lengths of the reviews in the dataset to know which is the 95% of the reviews have a length that is less than or equal to this value.



```
SEQUENCE_LEN = int(np.percentile(df.lengths, 95))
VOCAB_SIZE = 30000
vectorize_layer = TextVectorization(
    max_tokens=VOCAB_SIZE,
    standardize=standardization,
    output_mode="int",
    output_sequence_length=SEQUENCE_LEN)
```

```
[ ] SEQUENCE_LEN
```

```
161
```

Figure 4.18 : “95th percentile of the lengths” Code.

We found that 95% of the reviews are 161 tokens or fewer. Thus, each review will be padded to 161 tokens when processed by the `TextVectorization` layer.

3- Text data preprocessing :

We designed a function named `standardization` to preprocess and clean text reviews. It performs several steps to standardize the text, making it easier to work with in text processing.

Steps : 1- Convert to Lowercase.

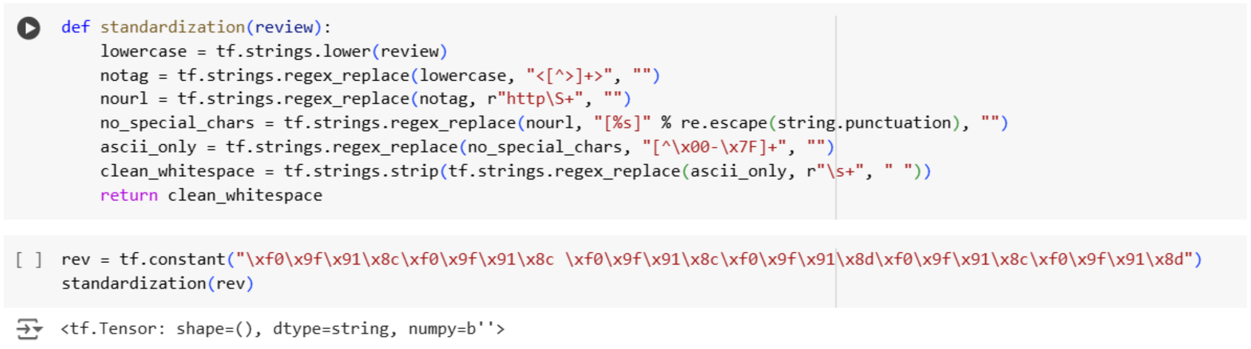
2- Remove HTML Tags.

3- Remove URLs.

4- Remove Special Characters.

5- Remove Non-ASCII Characters “like emoji characters”.

6- Clean Whitespace



```
def standardization(review):
    lowercase = tf.strings.lower(review)
    notag = tf.strings.regex_replace(lowercase, "<[^>]+>", "")
    nourl = tf.strings.regex_replace(notag, r"http\S+", "")
    no_special_chars = tf.strings.regex_replace(nourl, "[%s]" % re.escape(string.punctuation), "")
    ascii_only = tf.strings.regex_replace(no_special_chars, "[^\x00-\x7F]+", "")
    clean_whitespace = tf.strings.strip(tf.strings.regex_replace(ascii_only, r"\s+", " "))
    return clean_whitespace
```

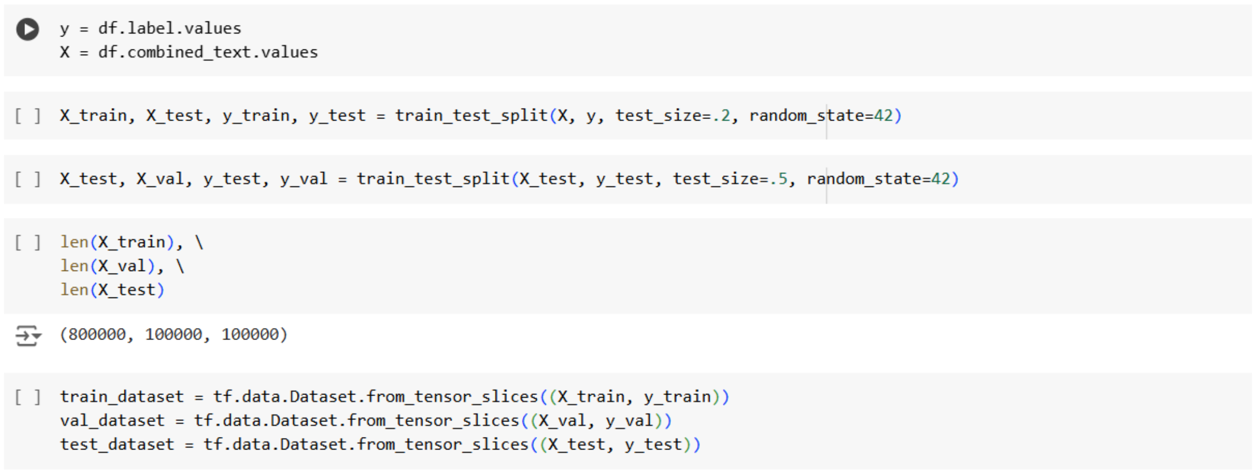
```
[ ] rev = tf.constant("\xf0\x9f\x91\x8c\xf0\x9f\x91\x8c \xf0\x9f\x91\x8c\xf0\x9f\x91\x8d\xf0\x9f\x91\x8c\xf0\x9f\x91\x8d")
    standardization(rev)
```

```
<tf.Tensor: shape=(), dtype=string, numpy=b''>
```

Figure 4.19 : “standardization function” Code.

4- Spliting the data :

We have 1,000,000 reviews in the dataset. We split them into three parts : 800,000 to train our models, 100,000 to test the models accuracy and 100,000 for validation.



```
y = df.label.values
X = df.combined_text.values
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2, random_state=42)
```

```
X_test, X_val, y_test, y_val = train_test_split(X_test, y_test, test_size=.5, random_state=42)
```

```
len(X_train), \
len(X_val), \
len(X_test)
```

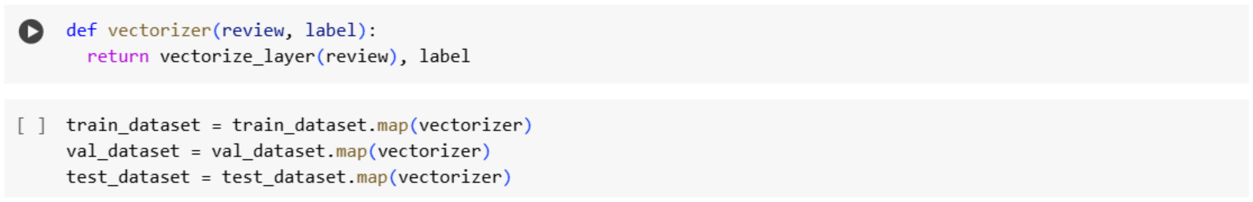
```
(800000, 100000, 100000)
```

```
train_dataset = tf.data.Dataset.from_tensor_slices((X_train, y_train))
val_dataset = tf.data.Dataset.from_tensor_slices((X_val, y_val))
test_dataset = tf.data.Dataset.from_tensor_slices((X_test, y_test))
```

Figure 4.20 : : “data split” Code.

5- Text Vectorization :

We designed a function named `vectorizer`. This function takes a review and its corresponding label as input and returns the vectorized representation of the review along with the label.



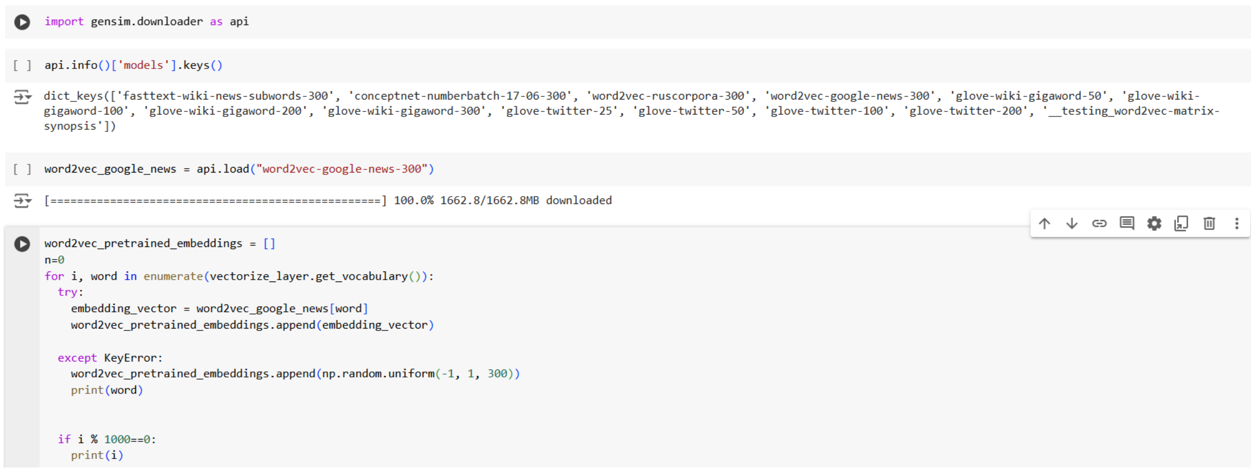
```
def vectorizer(review, label):
  return vectorize_layer(review), label
```

```
train_dataset = train_dataset.map(vectorizer)
val_dataset = val_dataset.map(vectorizer)
test_dataset = test_dataset.map(vectorizer)
```

Figure 4.21 : “vectorizer function” Code.

6- Integrating pre-trained Word2vec layer :

We create a list `word2vec_pretrained_embeddings` that stores word embedding either from a pre-trained word2vec model (`word2vec_google_news`) or random embedding if a word is not found in the pre-trained model.



```
import gensim.downloader as api
```

```
api.info()['models'].keys()
```

```
dict_keys(['fasttext-wiki-news-subwords-300', 'conceptnet-numberbatch-17-06-300', 'word2vec-ruscorpora-300', 'word2vec-google-news-300', 'glove-wiki-gigaword-50', 'glove-wiki-gigaword-100', 'glove-wiki-gigaword-200', 'glove-wiki-gigaword-300', 'glove-twitter-25', 'glove-twitter-50', 'glove-twitter-100', 'glove-twitter-200', '__testing_word2vec-matrix-synopsis'])
```

```
word2vec_google_news = api.load("word2vec-google-news-300")
```

```
[==================================================] 100.0% 1662.8/1662.8MB downloaded
```

```
word2vec_pretrained_embeddings = []
n=0
for i, word in enumerate(vectorize_layer.get_vocabulary()):
  try:
    embedding_vector = word2vec_google_news[word]
    word2vec_pretrained_embeddings.append(embedding_vector)

  except KeyError:
    word2vec_pretrained_embeddings.append(np.random.uniform(-1, 1, 300))
    print(word)


  if i % 1000==0:
    print(i)
```

Figure 4.22 : “embedding” Code.

4.5.1.3 Model building :

- LSTM model

  We initialize the early stop to Monitors the validation loss during training and stops the training process if the validation loss does not improve for a specified number of epochs. In LSTM Architecture :

  - Input Layer Defines the input shape as `(SEQUENCE_LEN,)`
  - in Embedding Layer Maps input tokens to dense vectors of size `EMBEDDING_DIMS`, with a vocabulary size of `VOCAB_SIZE`.
  - First LSTM Layer Contains 32 LSTM units, returns sequences for the next layer, and applies a dropout rate of 0.6 to prevent overfitting.
  - Second LSTM Layer Contains 16 LSTM units and returns the final hidden state for the sequence.
  - Dense Layer A single-unit output layer with a sigmoid activation function for binary classification.

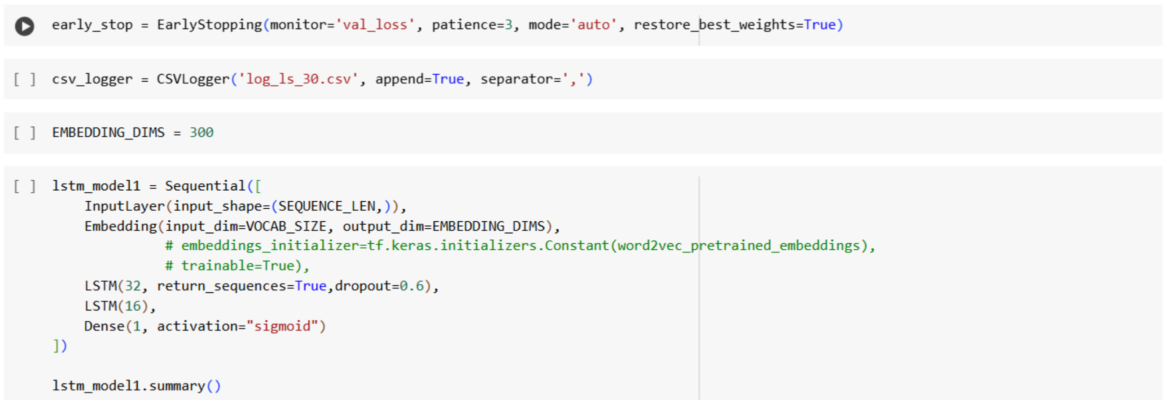
Why This Architecture?

We use one dense layer with a single neuron for feature classification. After reviewing various papers, we found that Random Forest often achieves high accuracy. This led us to consider combining deep learning models with machine learning models, using deep learning for feature extraction and machine learning for feature classification. However, this approach resulted in overfitting, with the machine learning models achieving 100% accuracy on the training set but only 60% accuracy on the validation set. Therefore, we opted for a very simple model for feature classification—a dense layer with one neuron.

We observed that higher dimensions for the embedding vector yielded better results, so we decided to use 300-dimensional embeddings. Although we used pretrained embedding vectors, the accuracy did not change significantly.

For the feature extractor models (sequence models), we did not use Conv1D because it did not perform well with long sequences. Instead, we chose between LSTM and GRU, and found that LSTM gave us better results.

Displays the model architecture, output shapes, and parameter counts, providing a comprehensive overview of the model's structure.



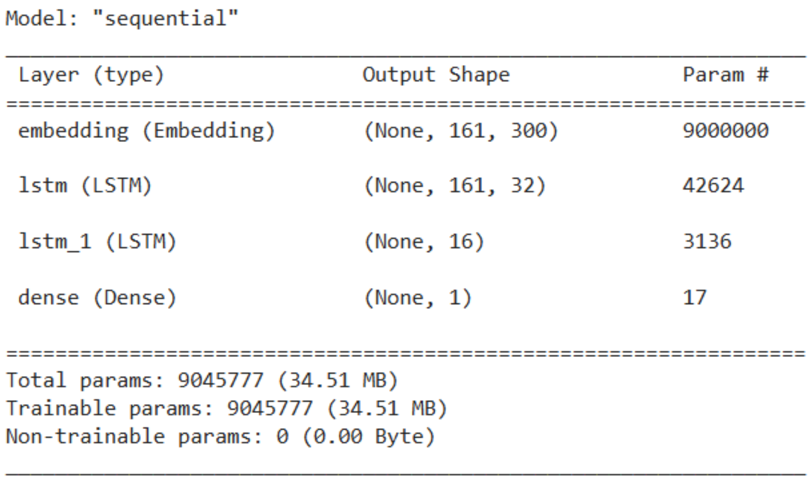
```
early_stop = EarlyStopping(monitor='val_loss', patience=3, mode='auto', restore_best_weights=True)
```

```
csv_logger = CSVLogger('log_ls_30.csv', append=True, separator=',')
```

```
EMBEDDING_DIMS = 300
```

```
lstm_model1 = Sequential([
    InputLayer(input_shape=(SEQUENCE_LEN,)),
    Embedding(input_dim=VOCAB_SIZE, output_dim=EMBEDDING_DIMS),
              # embeddings_initializer=tf.keras.initializers.Constant(word2vec_pretrained_embeddings),
              # trainable=True),
    LSTM(32, return_sequences=True,dropout=0.6),
    LSTM(16),
    Dense(1, activation="sigmoid")
])

lstm_model1.summary()
```

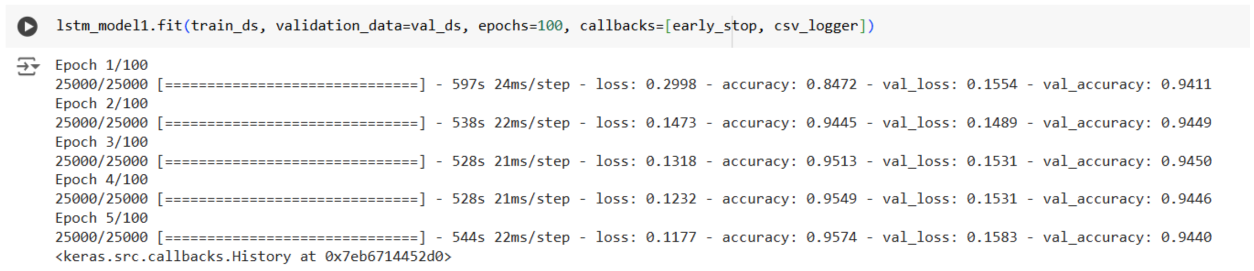
Figure 4.23 : “LSTM layers” Code.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 161, 300)          9000000

 lstm (LSTM)                 (None, 161, 32)           42624

 lstm_1 (LSTM)               (None, 16)                3136

 dense (Dense)               (None, 1)                 17

=================================================================
Total params: 9045777 (34.51 MB)
Trainable params: 9045777 (34.51 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 4.24 : LSTM layers summary.

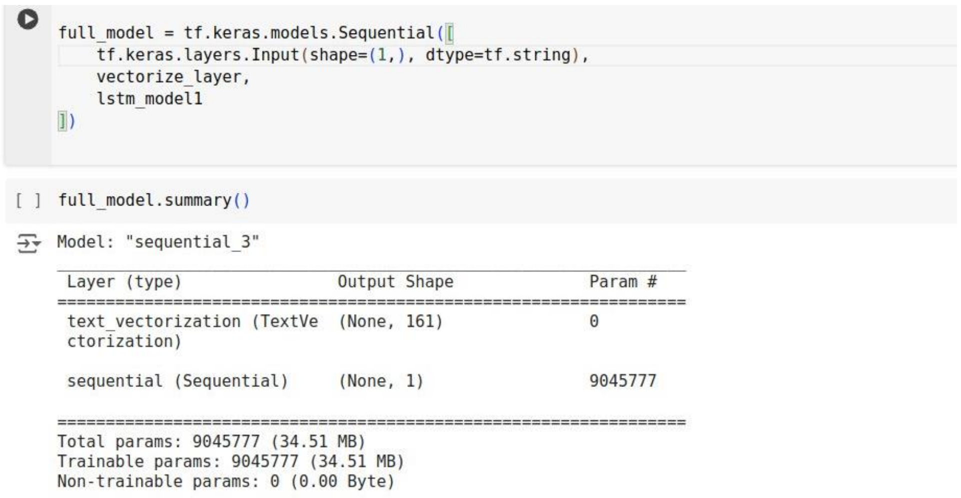
The LSTM model is trained using the `train_ds` dataset, with validation performed on the `val_ds` dataset. The training process is configured to run for up to 100 epochs, with early stopping and CSV logging callbacks to enhance training efficiency and facilitate performance tracking.



```
lstm_model1.fit(train_ds, validation_data=val_ds, epochs=100, callbacks=[early_stop, csv_logger])

Epoch 1/100
25000/25000 [==============================] - 597s 24ms/step - loss: 0.2998 - accuracy: 0.8472 - val_loss: 0.1554 - val_accuracy: 0.9411
Epoch 2/100
25000/25000 [==============================] - 538s 22ms/step - loss: 0.1473 - accuracy: 0.9445 - val_loss: 0.1489 - val_accuracy: 0.9449
Epoch 3/100
25000/25000 [==============================] - 528s 21ms/step - loss: 0.1318 - accuracy: 0.9513 - val_loss: 0.1531 - val_accuracy: 0.9450
Epoch 4/100
25000/25000 [==============================] - 528s 21ms/step - loss: 0.1232 - accuracy: 0.9549 - val_loss: 0.1531 - val_accuracy: 0.9446
Epoch 5/100
25000/25000 [==============================] - 544s 22ms/step - loss: 0.1177 - accuracy: 0.9574 - val_loss: 0.1583 - val_accuracy: 0.9440
<keras.src.callbacks.History at 0x7eb6714452d0>
```

Figure 4.25 : “LSTM training” Code.

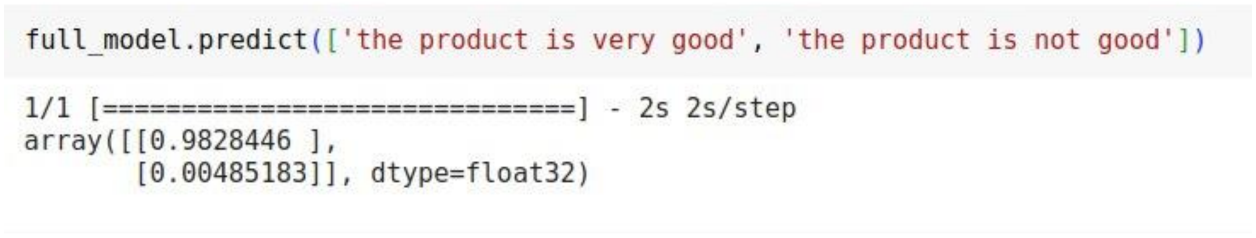
We make a combination from the LSTM model and vectorizer, then save it.



```
full_model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(1,), dtype=tf.string),
    vectorize_layer,
    lstm_model1
])
```

```
[ ] full_model.summary()

Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization (TextVe  (None, 161)               0
 ctorization)

 sequential (Sequential)     (None, 1)                 9045777

=================================================================
Total params: 9045777 (34.51 MB)
Trainable params: 9045777 (34.51 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.26 : Combination of LSTM and vectorizer.



```
full_model.predict(['the product is very good', 'the product is not good'])

1/1 [==============================] - 2s 2s/step
array([[0.9828446 ],
       [0.00485183]], dtype=float32)
```

Figure 4.27 : “test prediction” Code.

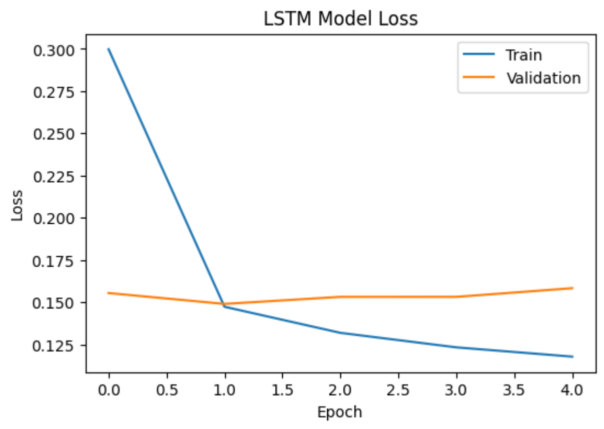


Figure 4.28 : LSTM model loss.

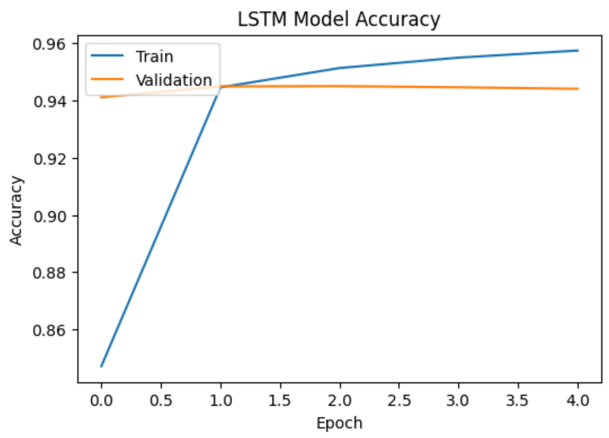
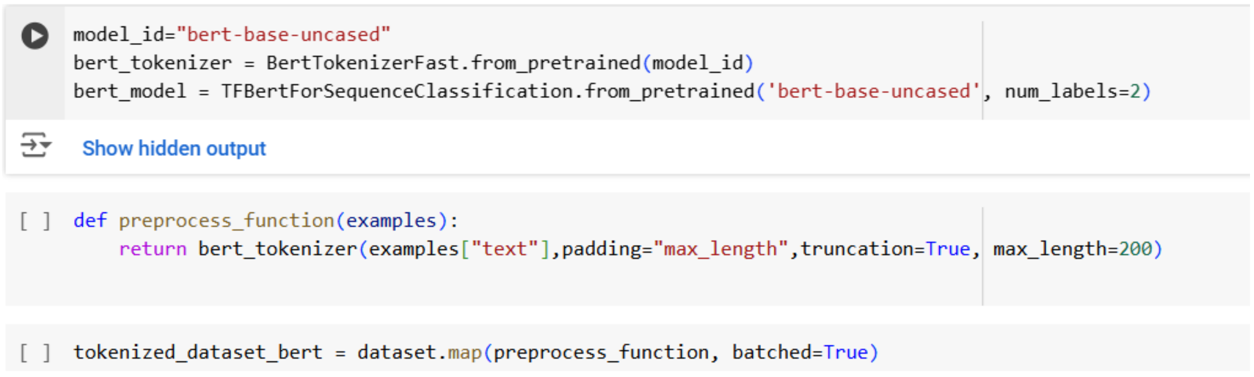


Figure 4.29 : LSTM model accuracy.

- BERT
  We initialize the BERT model and tokenizer `bert_tokenizer` and `bert_model` for sequence classification. The `bert_tokenizer` variable is initialized using `BertTokenizerFast.from_pretrained(model_id)`. This tokenizer is specifically designed for BERT models and facilitates fast tokenization of input text and the tokenizer is specifically designed for BERT models and facilitates fast tokenization of input text. We use a pre-trained BERT model configured for sequence classification tasks with two labels 2 for binary classification.



```
model_id="bert-base-uncased"
bert_tokenizer = BertTokenizerFast.from_pretrained(model_id)
bert_model = TFBertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2)
```

```
def preprocess_function(examples):
    return bert_tokenizer(examples["text"],padding="max_length",truncation=True, max_length=200)
```

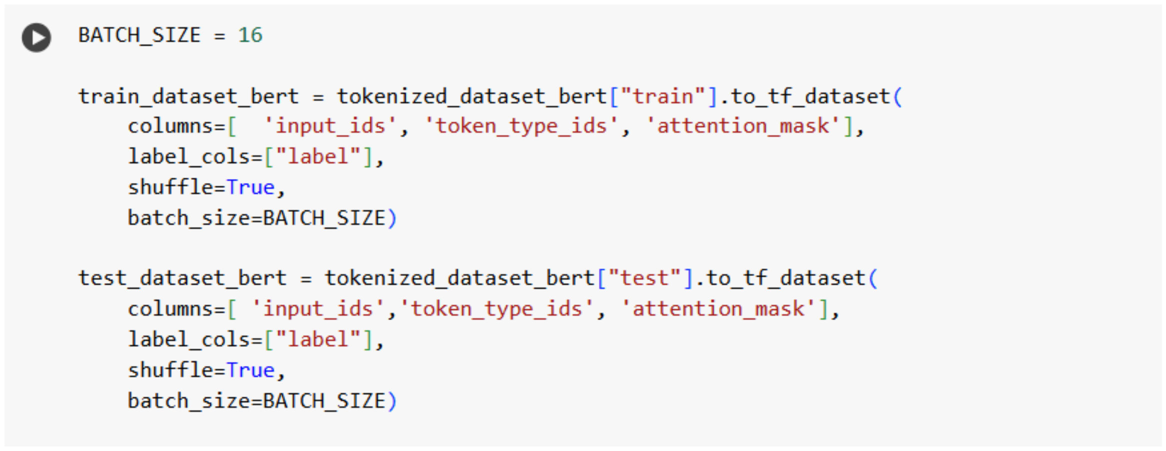
```
tokenized_dataset_bert = dataset.map(preprocess_function, batched=True)
```

Figure 4.30 : “BERT initialize” Code.

This setup allows for further fine-tuning of the BERT model on specific datasets or tasks such as sentiment analysis, by adjusting the training parameters and optimizer settings as needed.

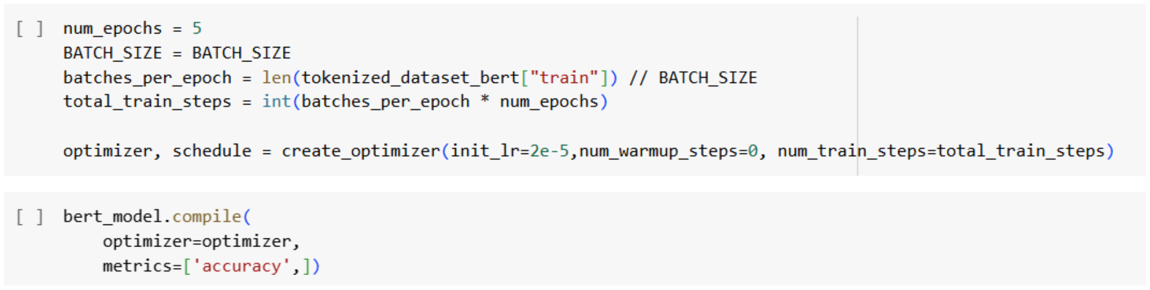
`BATCH_SIZE` determines the number of input samples processed together in each iteration of training during the training of a model.

The "train" split of `tokenized_dataset_bert` is designed for training the BERT model, containing batches of input sequences, token type IDs, and attention masks. We provide labels ensuring each batch includes corresponding labels for training purposes. The dataset is shuffled to randomize the order of samples in each epoch, enhancing model learning by reducing sequential dependencies.



```
BATCH_SIZE = 16

train_dataset_bert = tokenized_dataset_bert["train"].to_tf_dataset(
    columns=[  'input_ids', 'token_type_ids', 'attention_mask'],
    label_cols=["label"],
    shuffle=True,
    batch_size=BATCH_SIZE)

test_dataset_bert = tokenized_dataset_bert["test"].to_tf_dataset(
    columns=[ 'input_ids','token_type_ids', 'attention_mask'],
    label_cols=["label"],
    shuffle=True,
    batch_size=BATCH_SIZE)
```

Figure 4.31 : “BERT train & test data batches” Code.

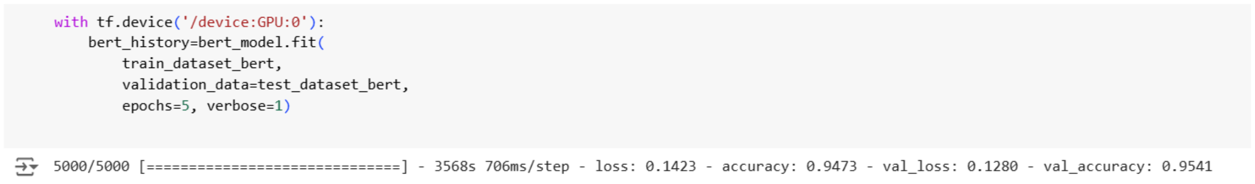


```
[ ] num_epochs = 5
    BATCH_SIZE = BATCH_SIZE
    batches_per_epoch = len(tokenized_dataset_bert["train"]) // BATCH_SIZE
    total_train_steps = int(batches_per_epoch * num_epochs)

    optimizer, schedule = create_optimizer(init_lr=2e-5,num_warmup_steps=0, num_train_steps=total_train_steps)
```

```
[ ] bert_model.compile(
        optimizer=optimizer,
        metrics=['accuracy',])
```

Figure 4.32 : “BERT compile” Code.

The training of the BERT model is configured to utilize GPU acceleration (`'/device:GPU:0'`). This speeds up computations, particularly beneficial for large-scale deep learning models like BERT.



```
with tf.device('/device:GPU:0'):
    bert_history=bert_model.fit(
        train_dataset_bert,
        validation_data=test_dataset_bert,
        epochs=5, verbose=1)
```

```
5000/5000 [==============================] - 3568s 706ms/step - loss: 0.1423 - accuracy: 0.9473 - val_loss: 0.1280 - val_accuracy: 0.9541
```

Figure 4.33 : “BERT training” Code.

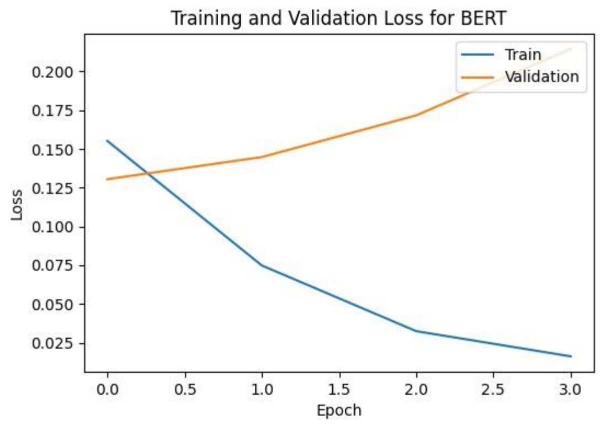


Figure 4.35 : BERT Loss.

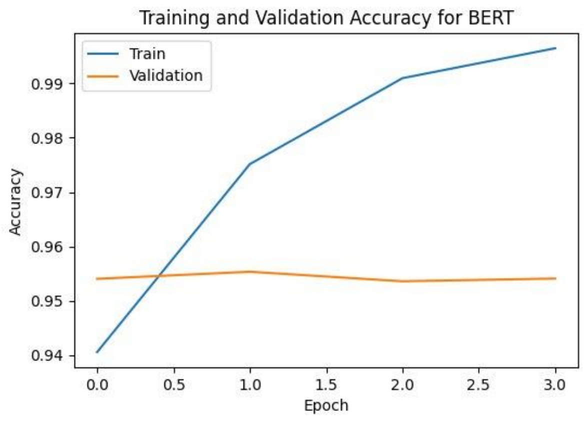
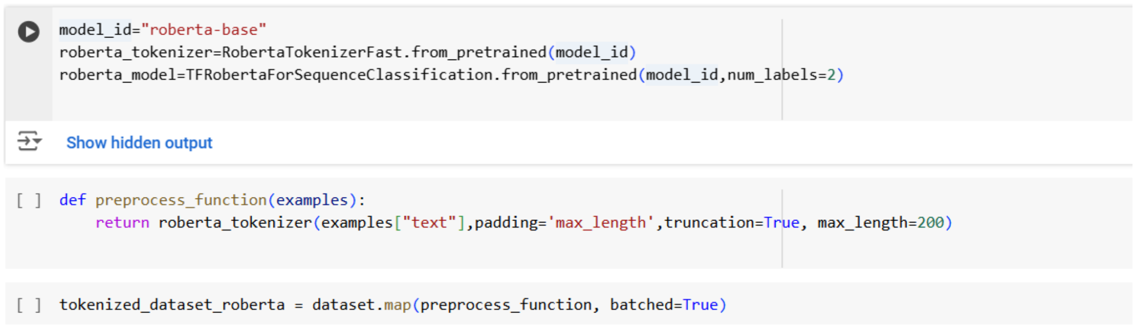


Figure 4.34 : BERT Accuracy.

- RoBERTa

We initialize the RoBERTa model and tokenizer variable using for sequence classification. The `RobertaTokenizerFast.from_pretrained(model_id) is optimized for RoBERTa models, efficiently tokenizes input text for subsequent processing.



```
model_id="roberta-base"
roberta_tokenizer=RobertaTokenizerFast.from_pretrained(model_id)
roberta_model=TFRobertaForSequenceClassification.from_pretrained(model_id,num_labels=2)
```

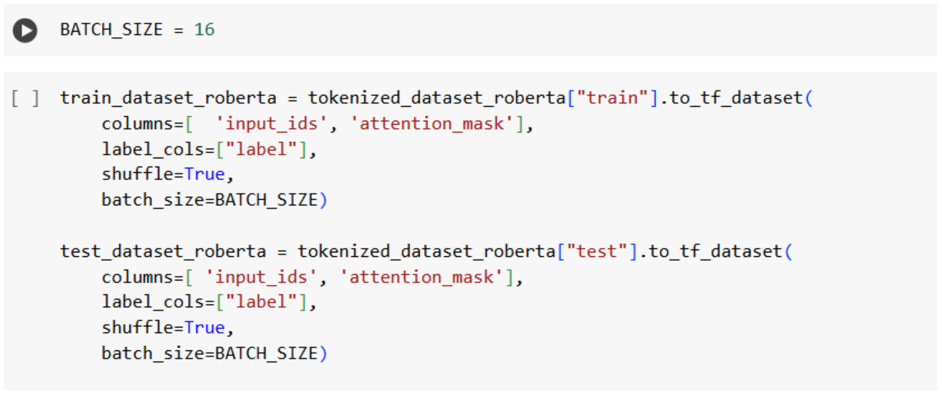
```
def preprocess_function(examples):
    return roberta_tokenizer(examples["text"],padding='max_length',truncation=True, max_length=200)
```

```
tokenized_dataset_roberta = dataset.map(preprocess_function, batched=True)
```

Figure 4.36 : “RoBERT initialize” Code.

As same as the BERT model steps, we implement the RoBERTa model. `BATCH_SIZE` determines the number of input samples processed together in each iteration of training during the training of a model.

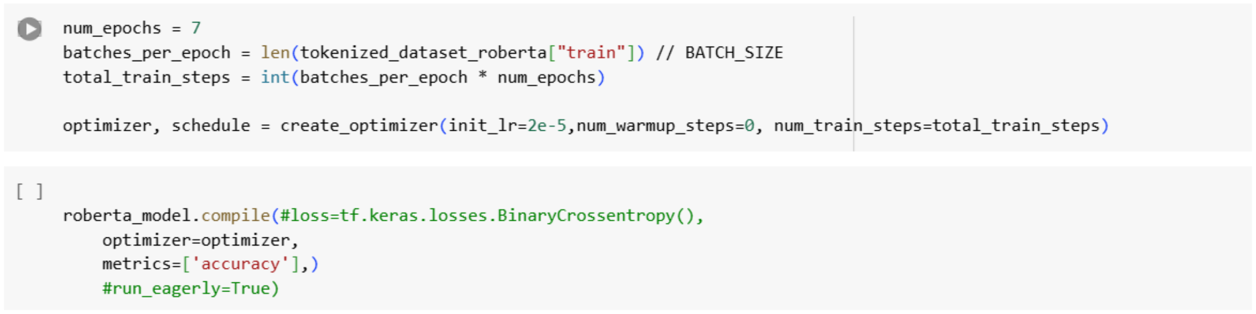
The "train" split of `tokenized_dataset_bert` is designed for training the BERT model, containing batches of input sequences, token type IDs, and attention masks. We provide labels ensuring each batch includes corresponding labels for training purposes. The dataset is shuffled to randomize the order of samples in each epoch, enhancing model learning by reducing sequential dependencies.



```
BATCH_SIZE = 16
```

```
train_dataset_roberta = tokenized_dataset_roberta["train"].to_tf_dataset(
    columns=[  'input_ids', 'attention_mask'],
    label_cols=["label"],
    shuffle=True,
    batch_size=BATCH_SIZE)

test_dataset_roberta = tokenized_dataset_roberta["test"].to_tf_dataset(
    columns=[ 'input_ids', 'attention_mask'],
    label_cols=["label"],
    shuffle=True,
    batch_size=BATCH_SIZE)
```

Figure 4.37 : “RoBERTa train & test data batches” Code.

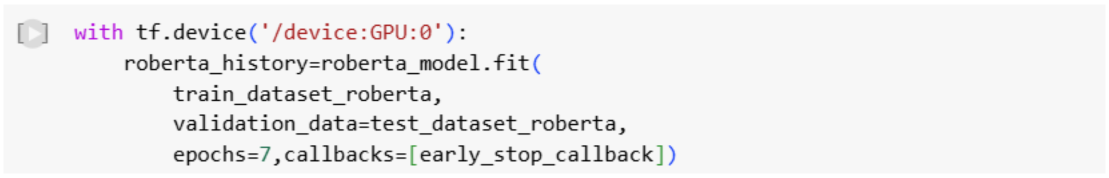


```
num_epochs = 7
batches_per_epoch = len(tokenized_dataset_roberta["train"]) // BATCH_SIZE
total_train_steps = int(batches_per_epoch * num_epochs)

optimizer, schedule = create_optimizer(init_lr=2e-5,num_warmup_steps=0, num_train_steps=total_train_steps)
```

```
roberta_model.compile(#loss=tf.keras.losses.BinaryCrossentropy(),
    optimizer=optimizer,
    metrics=['accuracy'],)
    #run_eagerly=True)
```

Figure 4.38 : “RoBERTa compile” Code.

The training of the RoBERTa model is configured to utilize GPU acceleration (`'/device:GPU:0'`). This speeds up computations, particularly beneficial for large-scale deep learning models like RoBERTa.



```
with tf.device('/device:GPU:0'):
    roberta_history=roberta_model.fit(
        train_dataset_roberta,
        validation_data=test_dataset_roberta,
        epochs=7,callbacks=[early_stop_callback])
```

Figure 4.39 : “RoBERTa training” Code.

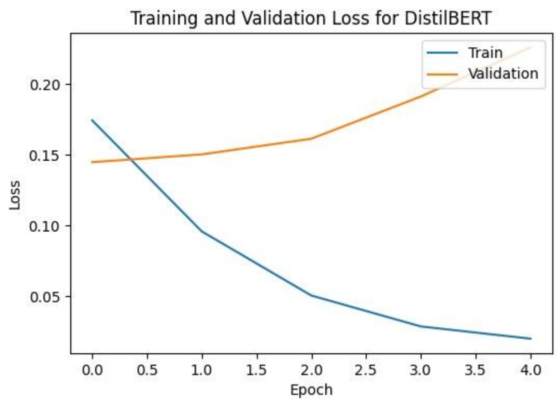


Figure 4.40 : RoBERTa loss.

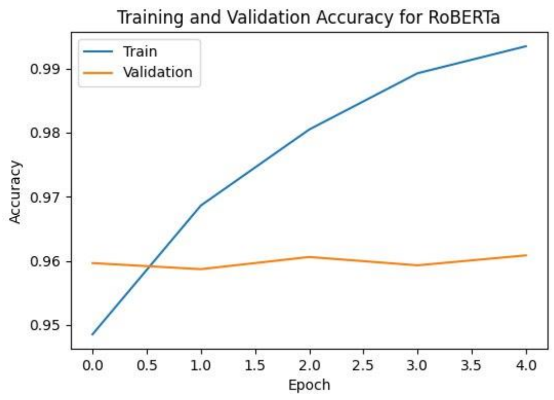
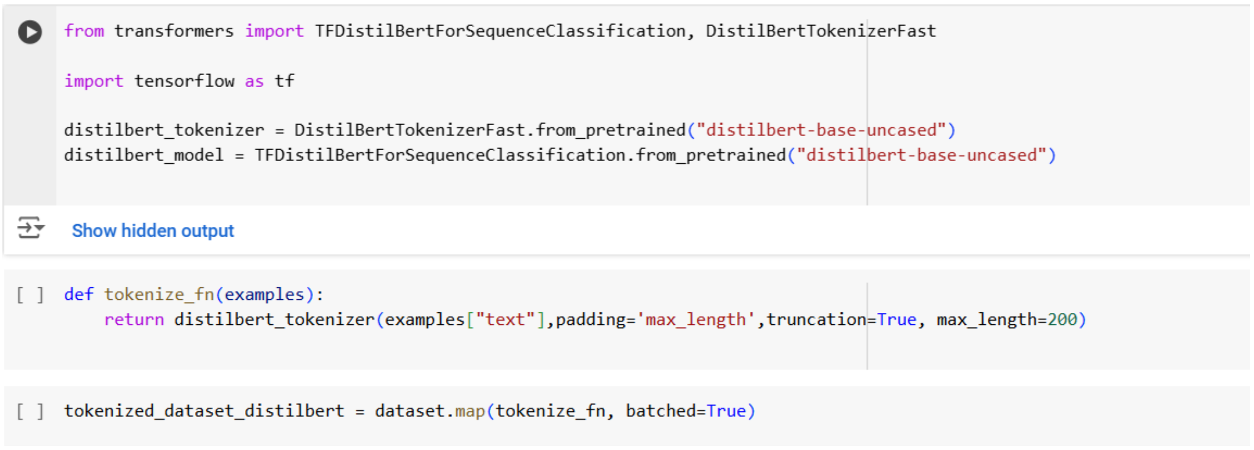


Figure 4.41 : RoBERTa Accuracy.

- DistilBERT
  For DistilBERT, the initialization and usage are also quite similar to BERT and RoBERTa. We initialize the BERT model and tokenizer `bert_tokenizer` and `bert_model` for sequence classification. The `bert_tokenizer` variable is initialized using `BertTokenizerFast.from_pretrained(model_id)`. This tokenizer is specifically designed for BERT models and facilitates fast tokenization of input text and the tokenizer is specifically designed for BERT models and facilitates fast tokenization of input text.



```
from transformers import TFDistilBertForSequenceClassification, DistilBertTokenizerFast

import tensorflow as tf

distilbert_tokenizer = DistilBertTokenizerFast.from_pretrained("distilbert-base-uncased")
distilbert_model = TFDistilBertForSequenceClassification.from_pretrained("distilbert-base-uncased")
```

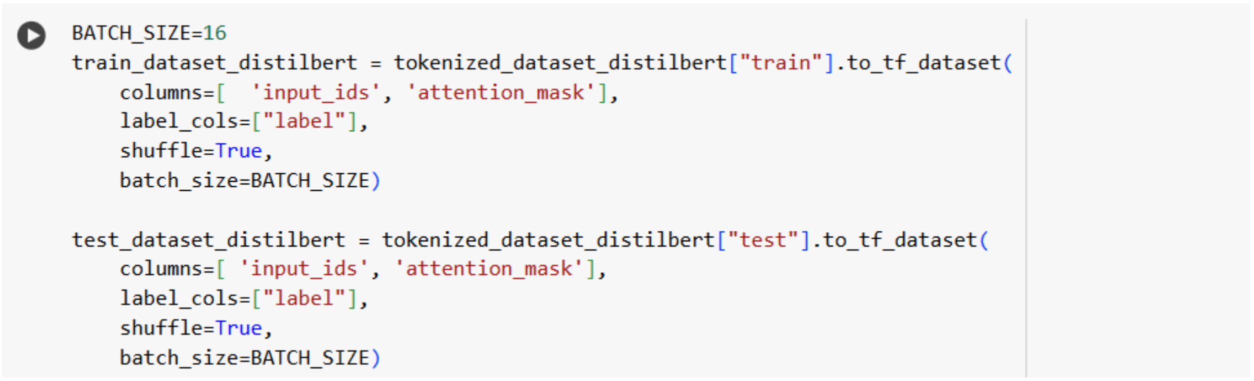
```
[ ] def tokenize_fn(examples):
        return distilbert_tokenizer(examples["text"],padding='max_length',truncation=True, max_length=200)
```

```
[ ] tokenized_dataset_distilbert = dataset.map(tokenize_fn, batched=True)
```

Figure 4.42 : “DistilBERT initialize” Code.

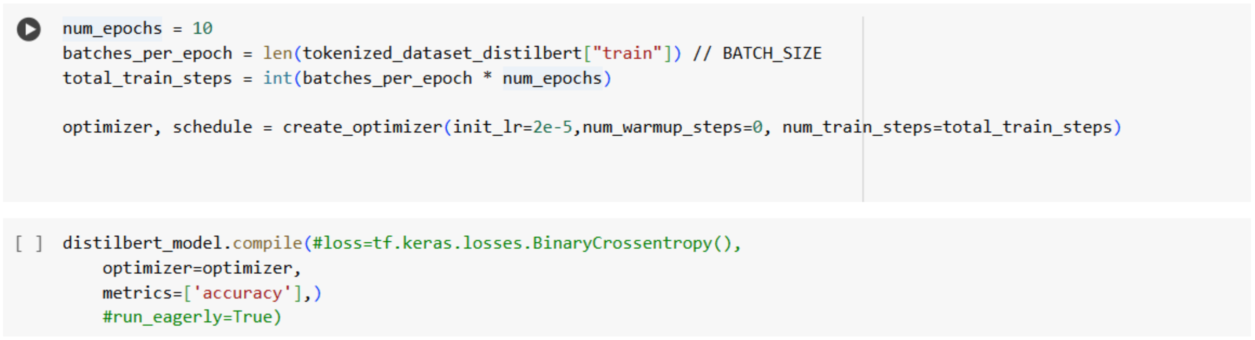
`BATCH_SIZE` determines the number of input samples processed together in each iteration of training during the training of a model.

The "train" split of `tokenized_dataset_bert` is designed for training the BERT model, containing batches of input sequences, token type IDs, and attention masks. We provide labels ensuring each batch includes corresponding labels for training purposes. The dataset is shuffled to randomize the order of samples in each epoch, enhancing model learning by reducing sequential dependencies.



```
BATCH_SIZE=16
train_dataset_distilbert = tokenized_dataset_distilbert["train"].to_tf_dataset(
    columns=[  'input_ids', 'attention_mask'],
    label_cols=["label"],
    shuffle=True,
    batch_size=BATCH_SIZE)

test_dataset_distilbert = tokenized_dataset_distilbert["test"].to_tf_dataset(
    columns=[ 'input_ids', 'attention_mask'],
    label_cols=["label"],
    shuffle=True,
    batch_size=BATCH_SIZE)
```

Figure 4.43 : “DistilBERT train & test data batches” Code.

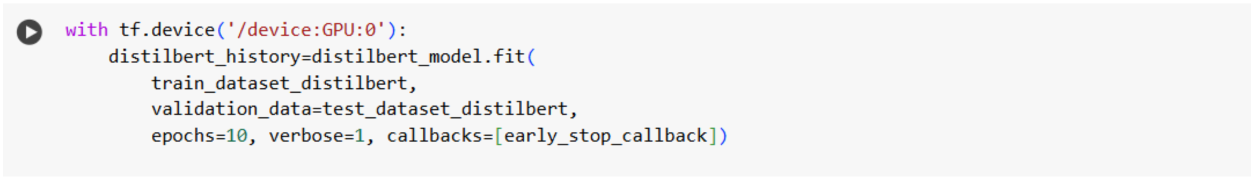


```
num_epochs = 10
batches_per_epoch = len(tokenized_dataset_distilbert["train"]) // BATCH_SIZE
total_train_steps = int(batches_per_epoch * num_epochs)

optimizer, schedule = create_optimizer(init_lr=2e-5,num_warmup_steps=0, num_train_steps=total_train_steps)
```

```
distilbert_model.compile(#loss=tf.keras.losses.BinaryCrossentropy(),
    optimizer=optimizer,
    metrics=['accuracy'],)
    #run_eagerly=True)
```

Figure 4.44 : “DistilBERT compile” Code.

The training of the RoBERT model is configured to utilize GPU acceleration (`'/device:GPU:0'`). This speeds up computations, particularly beneficial for large-scale deep learning models like RoBERT.



```
with tf.device('/device:GPU:0'):
    distilbert_history=distilbert_model.fit(
        train_dataset_distilbert,
        validation_data=test_dataset_distilbert,
        epochs=10, verbose=1, callbacks=[early_stop_callback])
```

Figure 4.45 : “DistilBERT training” Code.

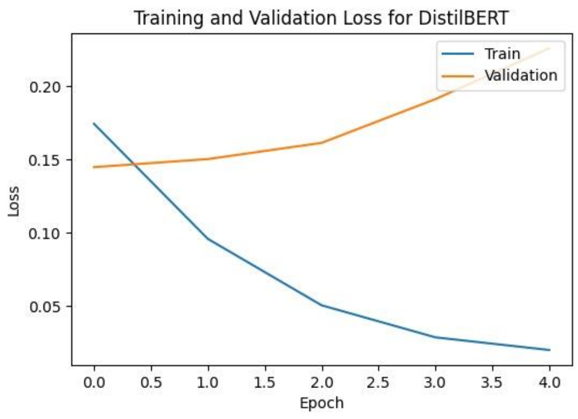


Figure 4.46 : DistilBERT Loss.

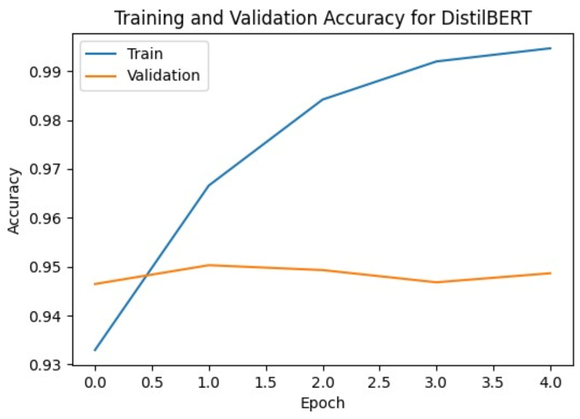
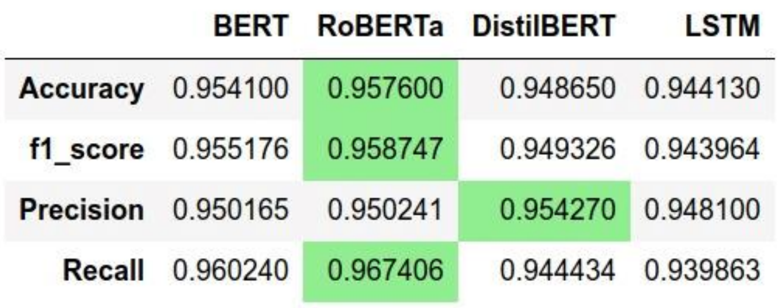


Figure 4.47 : DistilBERT Accuracy.

- Model Performance Comparison



	BERT	RoBERTa	DistilBERT	LSTM
Accuracy	0.954100	0.957600	0.948650	0.944130
f1_score	0.955176	0.958747	0.949326	0.943964
Precision	0.950165	0.950241	0.954270	0.948100
Recall	0.960240	0.967406	0.944434	0.939863

Figure 4.48 : Model Performance.

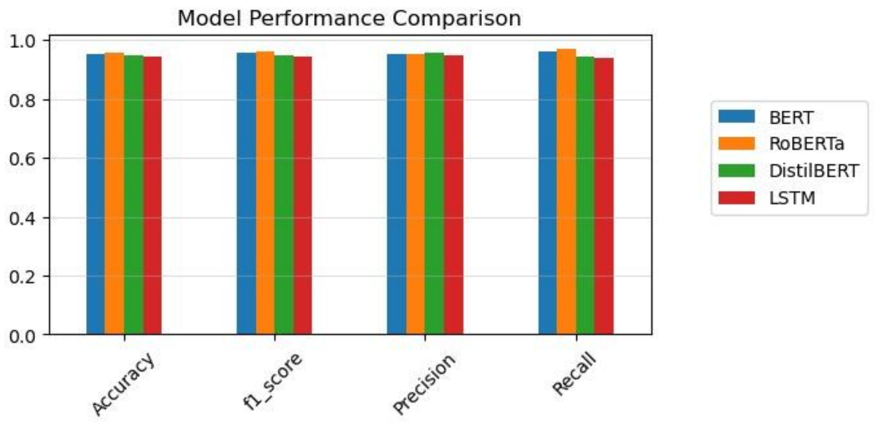
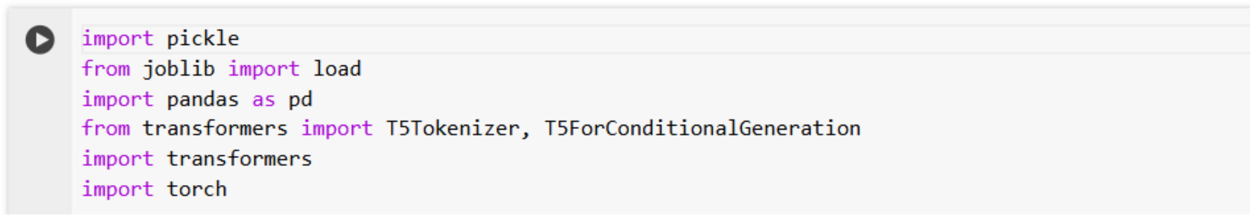


Figure 4.49 : Model Performance.

From the table above, we observe that RoBERTa has the highest accuracy. However, it is very slow in producing outputs because it requires a GPU. Even though we converted the model to ONNX format and applied dynamic quantization (which changes the dtype of the weights from Float32 to uint8, making the model lighter and faster), we ultimately decided to use an LSTM model. The difference in accuracy between RoBERTa and the LSTM model is minimal, but the LSTM model is significantly faster in generating outputs.

4.5.2 Summarization Models

4.5.2.1 Preparing the tools



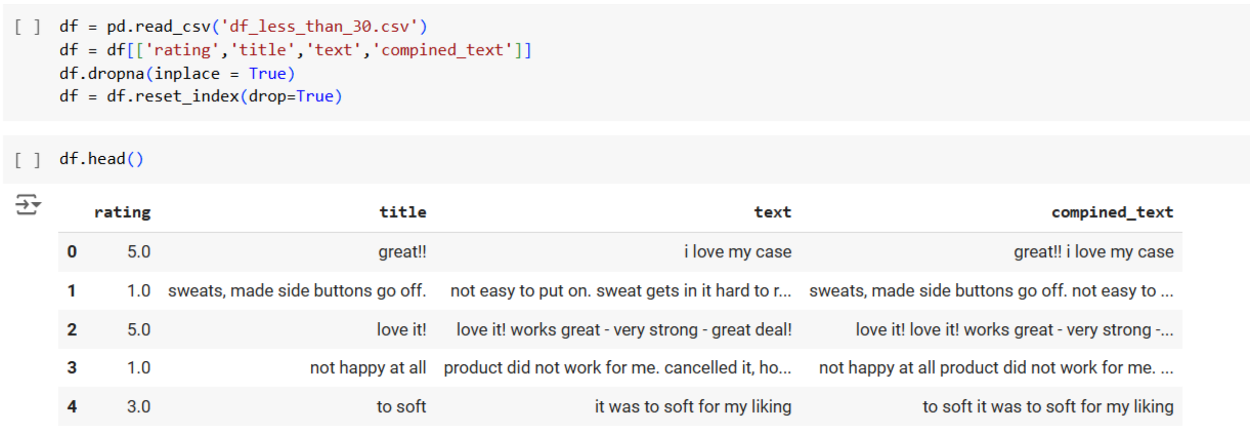
```
import pickle
from joblib import load
import pandas as pd
from transformers import T5Tokenizer, T5ForConditionalGeneration
import transformers
import torch
```

Figure 4.50 : “import libraries” Code.

4.5.2.2 Preparing the data

Here, we use only the reviews which has less than 30 sequence length for its reviews text. These are 600017 of the whole instances.

We select some features which we will need (rating, title, text, complained text). Depends on the rating we split the data to positive and negative reviews “all reviews have 3 or more in rating we considered it positive, otherwise is negative”. So, we get 300010 positive reviews and 300007 negative reviews.



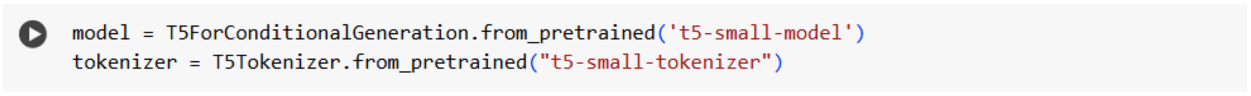
```
[ ] df = pd.read_csv('df_less_than_30.csv')
    df = df[['rating','title','text','compined_text']]
    df.dropna(inplace = True)
    df = df.reset_index(drop=True)

[ ] df.head()
```

	rating	title	text	compined_text
0	5.0	great!!	i love my case	great!! i love my case
1	1.0	sweats, made side buttons go off.	not easy to put on. sweat gets in it hard to r...	sweats, made side buttons go off. not easy to ...
2	5.0	love it!	love it! works great - very strong - great deal!	love it! love it! works great - very strong -...
3	1.0	not happy at all	product did not work for me. cancelled it, ho...	not happy at all product did not work for me. ...
4	3.0	to soft	it was to soft for my liking	to soft it was to soft for my liking

Figure 4.51 : “data load” Code.

4.5.2.3 Model Building

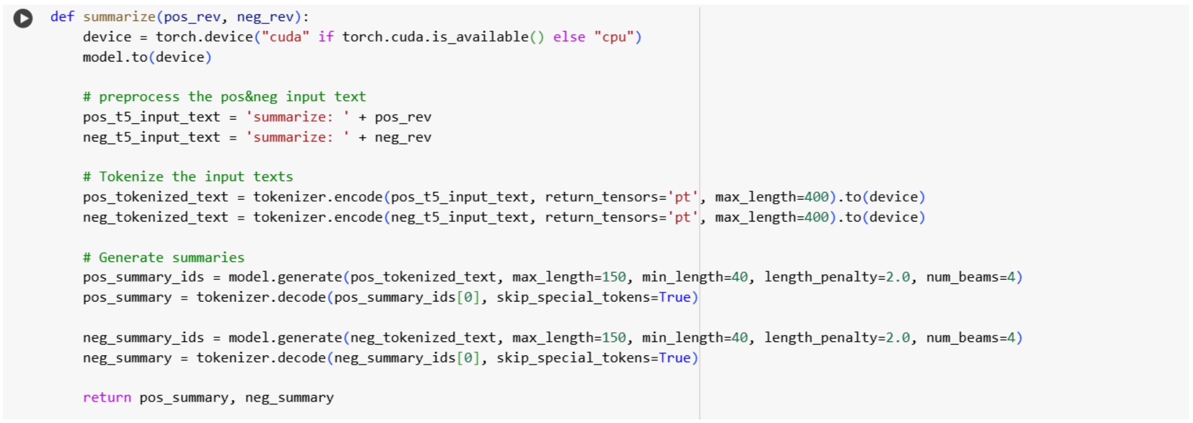
- T5-small
  We initialize a pre-trained T5 (Text-to-Text Transfer Transformer) model, specifically using the `t5-small` variant. The T5 model is designed for various text-to-text tasks, including summarization. The `t5-small` variant is a compact and efficient version of the T5 model, making it suitable for tasks that require fewer computational resources.



```python
model = T5ForConditionalGeneration.from_pretrained('t5-small-model')
tokenizer = T5Tokenizer.from_pretrained("t5-small-tokenizer")
```

Figure 4.52 : “T5-small initialization” Code.

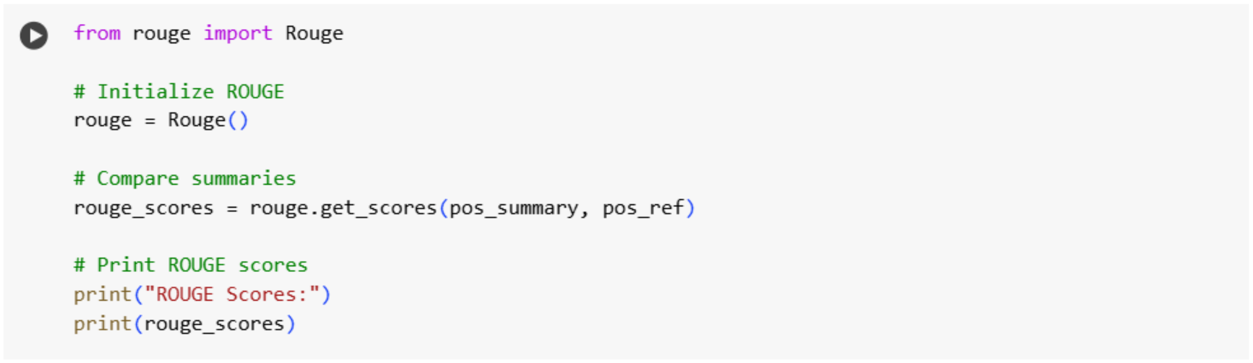
The `summarize` function takes positive and negative reviews as input and generates concise summaries for each using a pre-trained T5 model. This function leverages the capabilities of the T5 model for text summarization. Generates summaries for the tokenized reviews, with specified maximum and minimum lengths, length penalty, and number of beams for beam search.



```python
def summarize(pos_rev, neg_rev):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model.to(device)

    # preprocess the pos&neg input text
    pos_t5_input_text = 'summarize: ' + pos_rev
    neg_t5_input_text = 'summarize: ' + neg_rev

    # Tokenize the input texts
    pos_tokenized_text = tokenizer.encode(pos_t5_input_text, return_tensors='pt', max_length=400).to(device)
    neg_tokenized_text = tokenizer.encode(neg_t5_input_text, return_tensors='pt', max_length=400).to(device)

    # Generate summaries
    pos_summary_ids = model.generate(pos_tokenized_text, max_length=150, min_length=40, length_penalty=2.0, num_beams=4)
    pos_summary = tokenizer.decode(pos_summary_ids[0], skip_special_tokens=True)

    neg_summary_ids = model.generate(neg_tokenized_text, max_length=150, min_length=40, length_penalty=2.0, num_beams=4)
    neg_summary = tokenizer.decode(neg_summary_ids[0], skip_special_tokens=True)

    return pos_summary, neg_summary
```

Figure 4.53 : “summarization function” Code.

We use the complained text feature to test the generated summarization and “text” feature as a summarization reference. Using the ROUGE metric, we test the summarization model efficiency.



```python
from rouge import Rouge

# Initialize ROUGE
rouge = Rouge()

# Compare summaries
rouge_scores = rouge.get_scores(pos_summary, pos_ref)

# Print ROUGE scores
print("ROUGE Scores:")
print(rouge_scores)
```

Figure 4.54 : “ROUGE metric” Code.

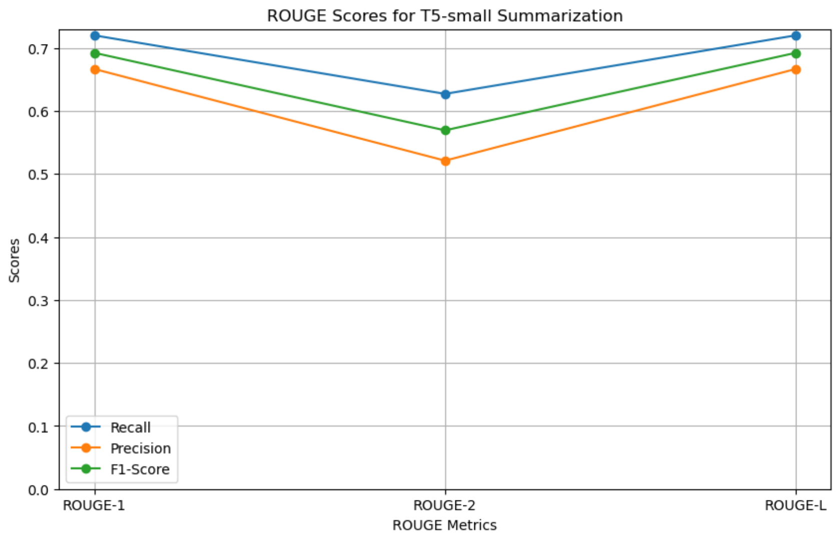
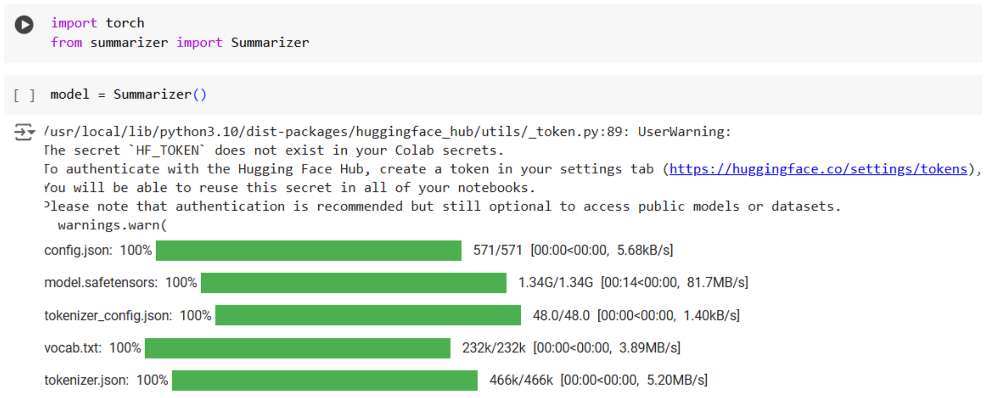


Figure 4.55 : ROUGE Scores for T5-small summarization.

- BERT
  We initialize a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model.. The BERT model is designed for various NLP tasks, including text summarization.



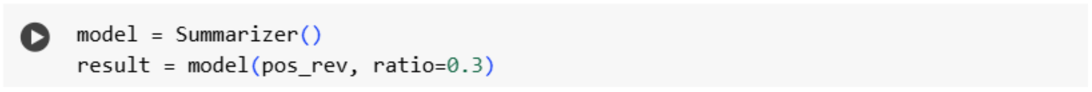
```
import torch
from summarizer import Summarizer
```

```
[ ] model = Summarizer()
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens),
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100%  571/571 [00:00<00:00, 5.68kB/s]
model.safetensors: 100%  1.34G/1.34G [00:14<00:00, 81.7MB/s]
tokenizer_config.json: 100%  48.0/48.0 [00:00<00:00, 1.40kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 3.89MB/s]
tokenizer.json: 100%  466k/466k [00:00<00:00, 5.20MB/s]
```

Figure 4.56 : “BERT initialize” Code.

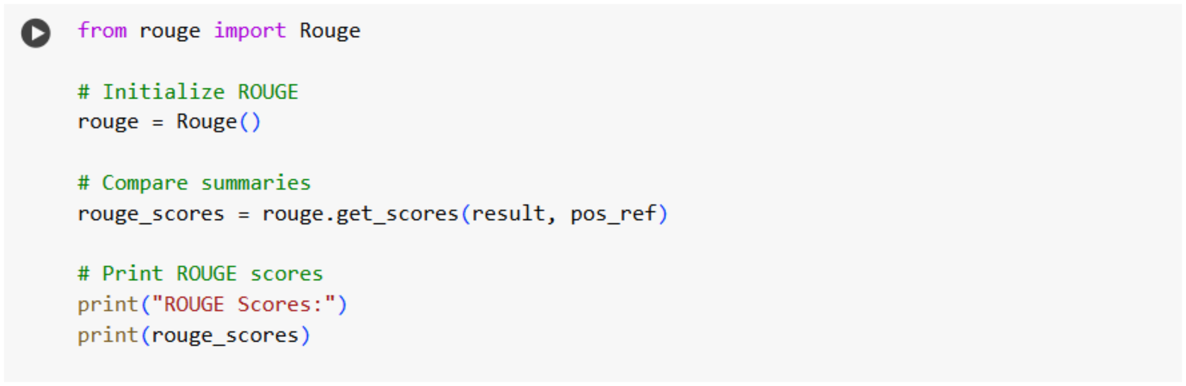
We utilize the summarizer library to perform text summarization with a pre-trained BERT model. The Summarizer class provides an easy-to-use interface for generating summaries with the specified ratio of 0.3. This means the summary will be approximately 30% the length of the original text.



```
model = Summarizer()
result = model(pos_rev, ratio=0.3)
```

Figure 4.57 : “BERT summarization” Code.

Same as T5 model, we use “text” feature as a summarization reference. Using the ROUGE metric, we test the summarization model efficiency.



```
from rouge import Rouge

# Initialize ROUGE
rouge = Rouge()

# Compare summaries
rouge_scores = rouge.get_scores(result, pos_ref)

# Print ROUGE scores
print("ROUGE Scores:")
print(rouge_scores)
```

Figure 4.58 : “ROUGE metric” Code.

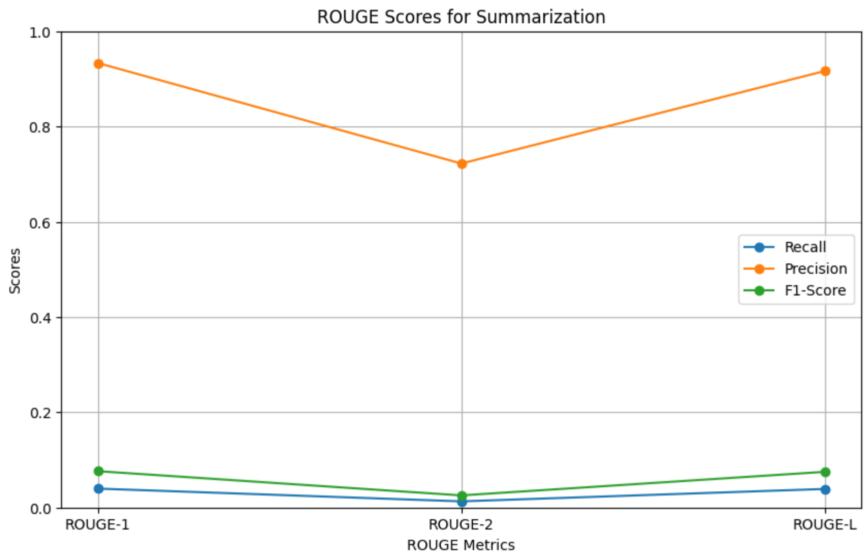


Figure 4.59 : “ROUGE metric” Code.

4.6 Discussion of results :

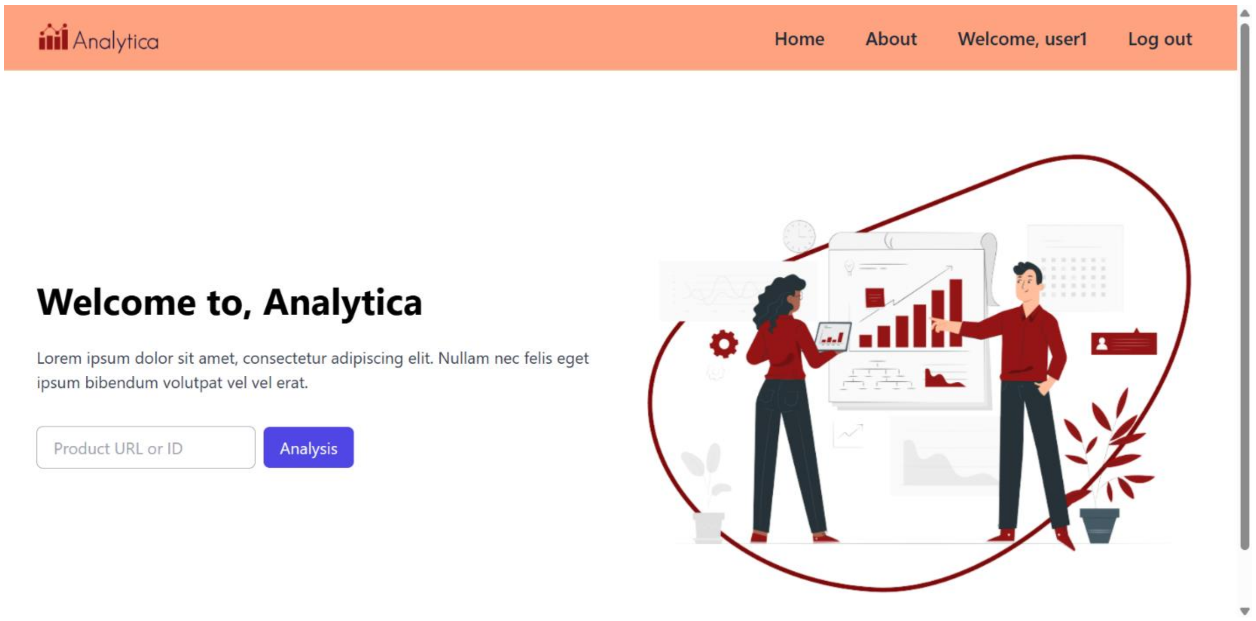
The final learning model we choose, a Transformer-based model, is RoBERTa,which is successfully classifies the reviews with an accuracy of 95.7%. it is excellent in understanding and classifying reviews. It is effective in predicting sentiment or other classifications from text.

For summarization we choose T5-small ,which is generate a good and clear summarization. It achieved a recall 0.72, precision of 0.667 and f1-score of 0.692 in evaluation metrics. The ability of T5-small to minimize large amounts of information into brief summaries makes it suitable for us to use in our project.

The assembly of these models enhanced the capabilities of our project through the use of natural language processing techniques, ensuring strong performance in both classification and summarization tasks.

4.7 UI/UX Design

4.7.1 Home Page

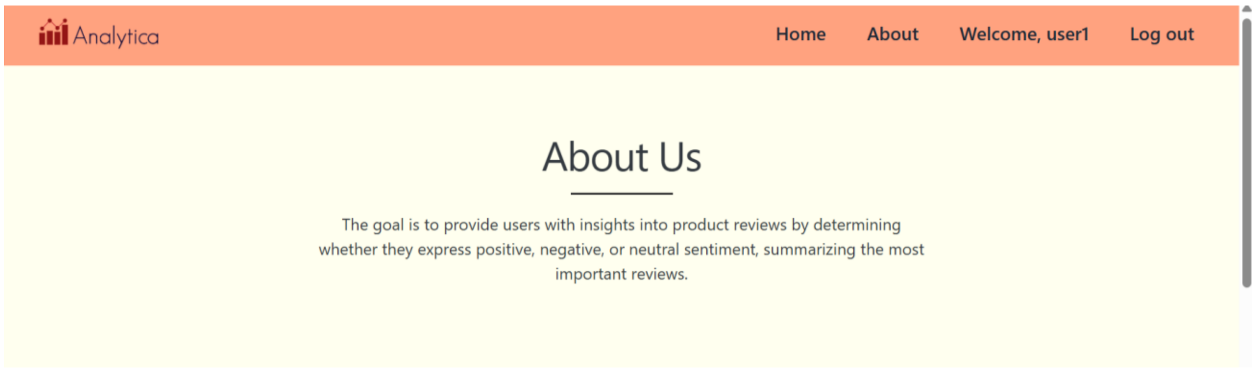


Welcome to, Analytica

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam nec felis eget ipsum bibendum volutpat vel vel erat.

Figure 4.60 : Home page.

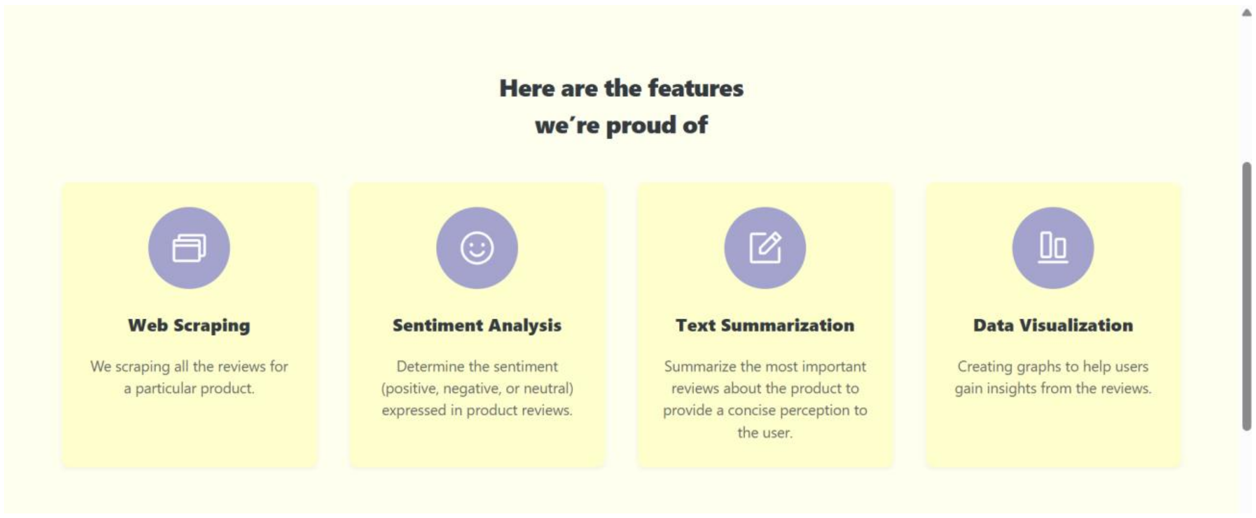
4.7.2 About us Page



About Us

The goal is to provide users with insights into product reviews by determining whether they express positive, negative, or neutral sentiment, summarizing the most important reviews.

Figure 4.61 : About us Page.


Here are the features we´re proud of

Web Scraping

We scraping all the reviews for a particular product.

Sentiment Analysis

Determine the sentiment (positive, negative, or neutral) expressed in product reviews.

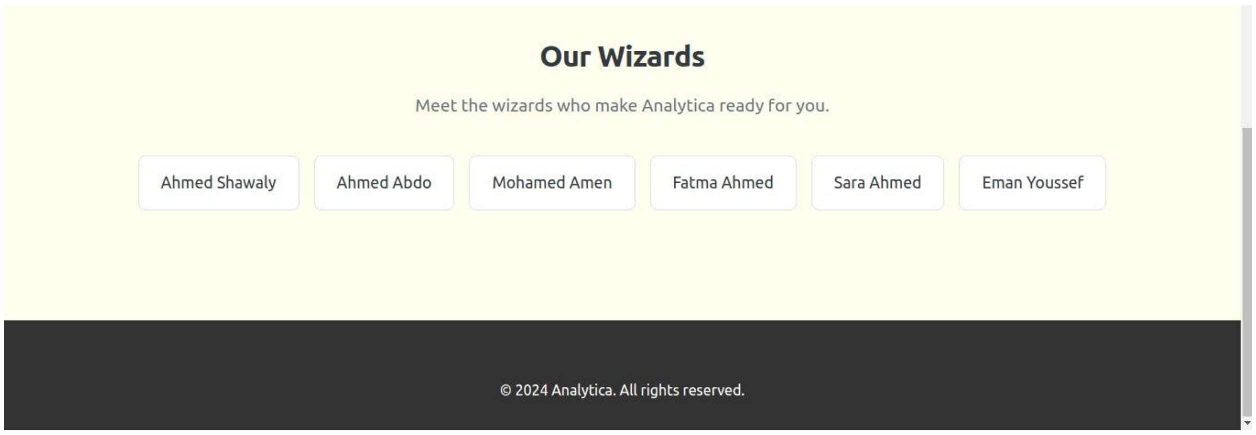
Text Summarization

Summarize the most important reviews about the product to provide a concise perception to the user.

Data Visualization

Creating graphs to help users gain insights from the reviews.

Figure 4.62 : Cont. About us Page.

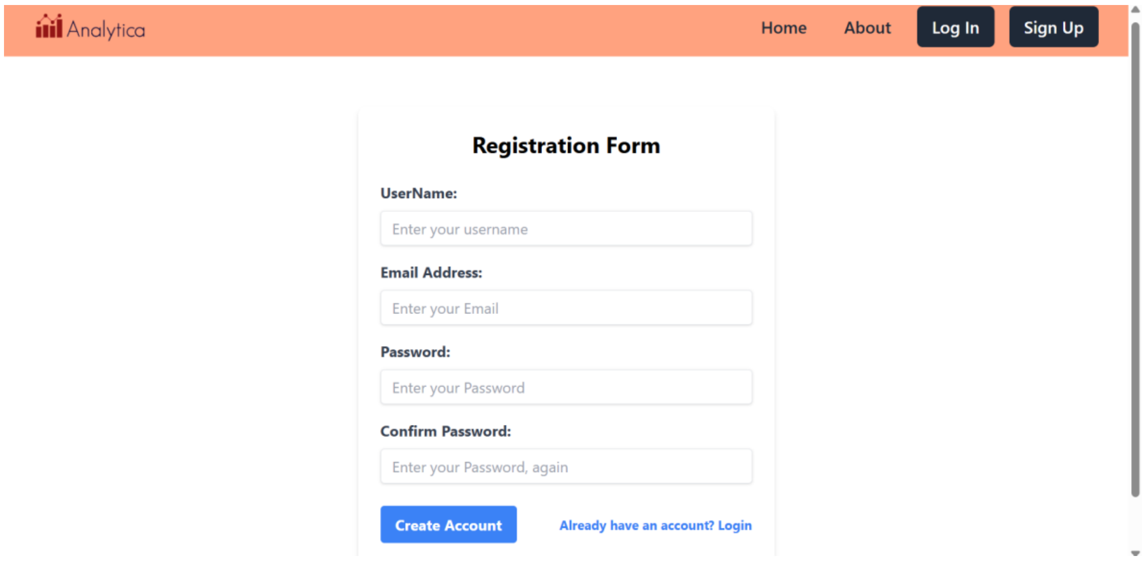

Our Wizards

Meet the wizards who make Analytica ready for you.

© 2024 Analytica. All rights reserved.

Figure 4.63 : Cont. About us Page.

4.7.3 Sign Up Page



Registration Form

UserName:

Enter your username

Email Address:

Enter your Email

Password:

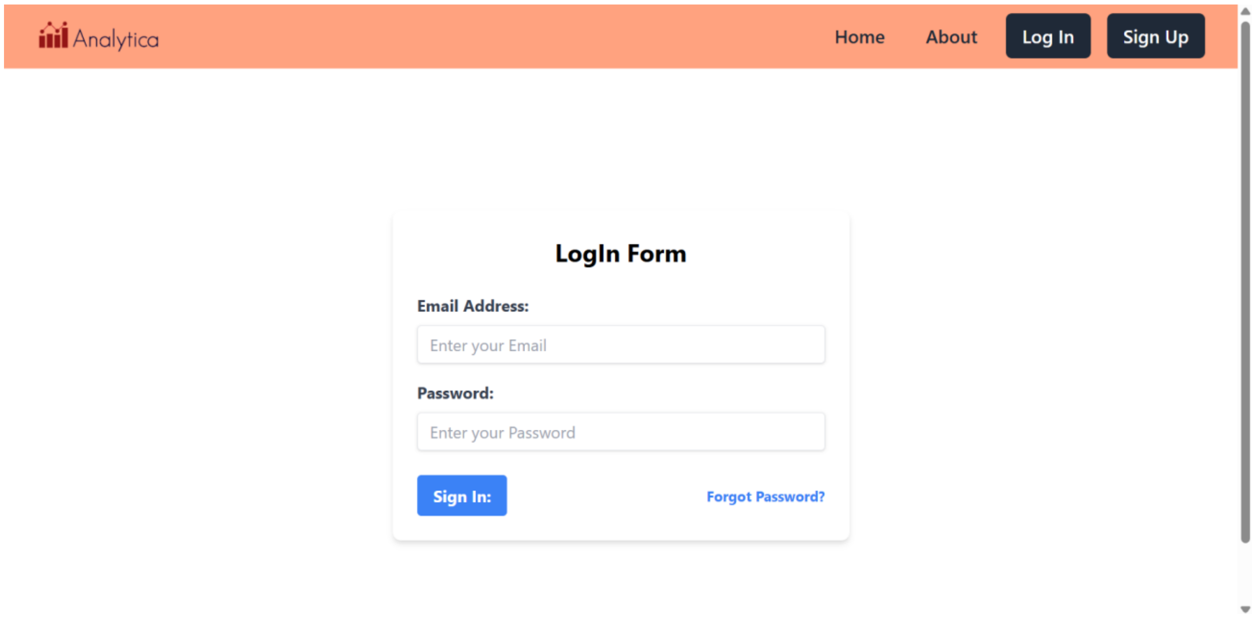
Enter your Password

Confirm Password:

Enter your Password, again

Figure 4.64 : Sign Up Page.

4.7.4 Login Page



LogIn Form

Email Address:

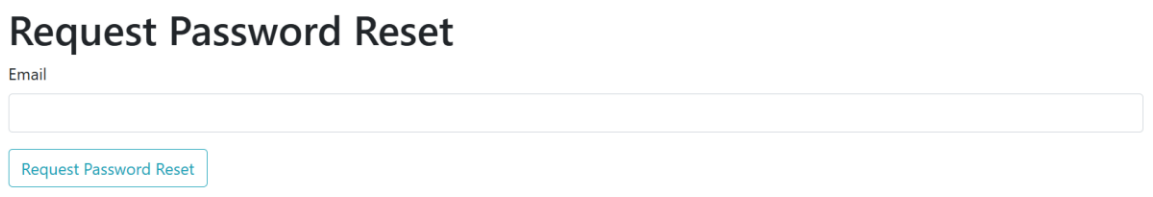
Enter your Email

Password:

Enter your Password

Figure 4.65 : Login Page.

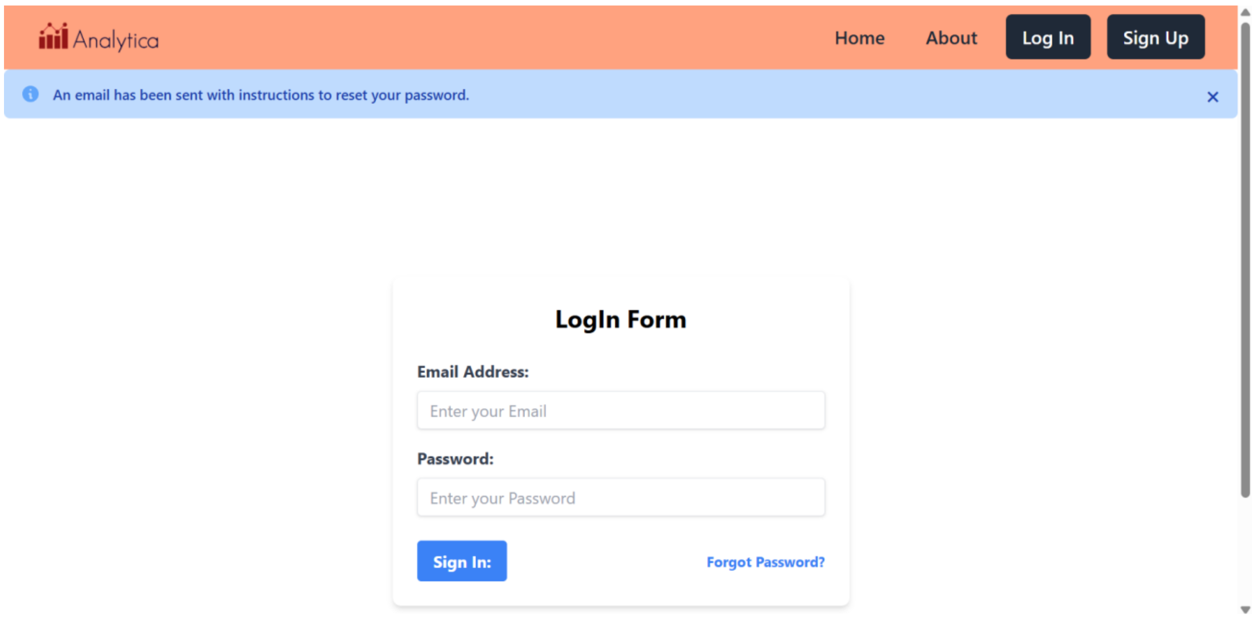
4.7.5 Request Password Page



Request Password Reset

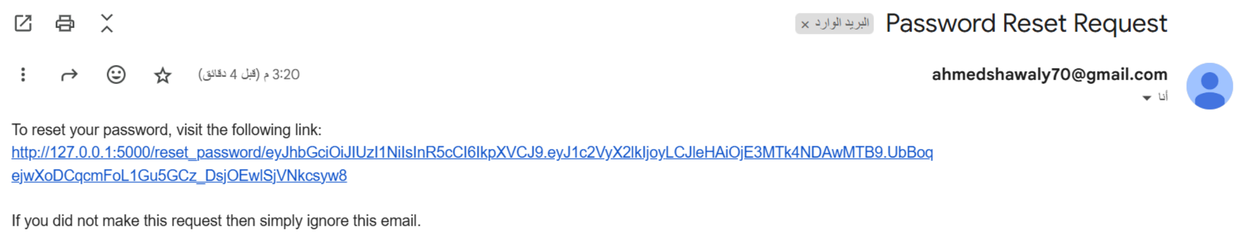
Email

Figure 4.66 : Request Password Page.



An email has been sent with instructions to reset your password.

Figure 4.67 : Cont. Request Password Page.



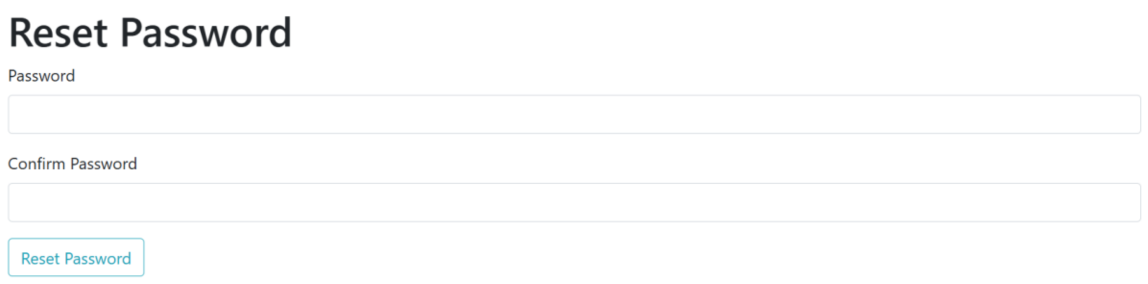
Password Reset Request

ahmedshawaly70@gmail.com

To reset your password, visit the following link:
http://127.0.0.1:5000/reset_password/eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoyLCJleHAiOjE3MTk4NDAwMTB9.UbBoq
ejwXoDCqcmFoL1Gu5GCz_DsjOEwlSjVNkcsyw8

If you did not make this request then simply ignore this email.

Figure 4.68 : Verification email.



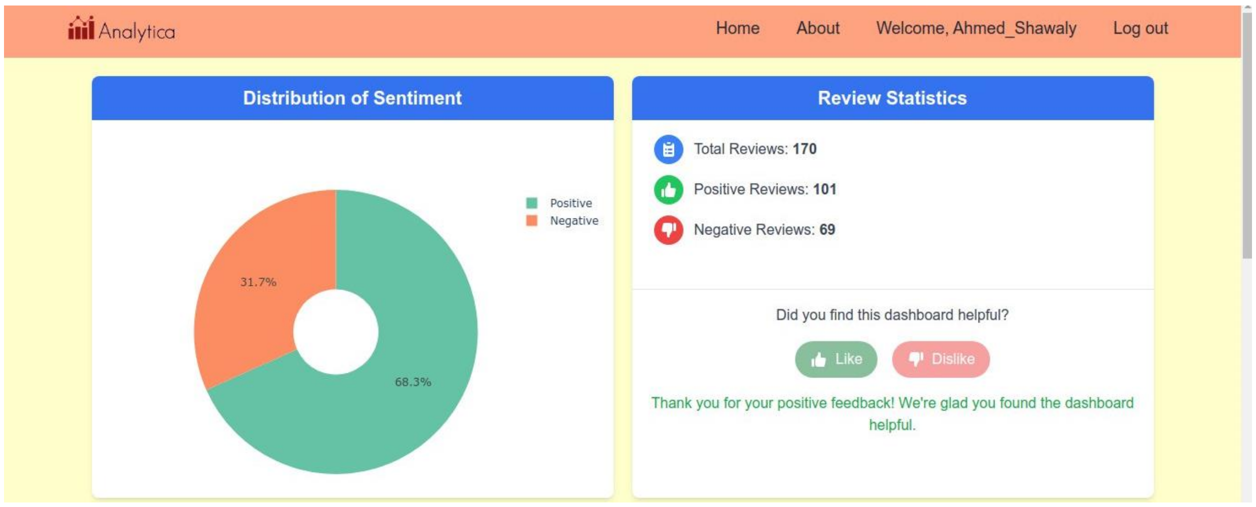
Reset Password

Password

Confirm Password

Figure 4.69 : Reset Password.

4.7.6 Result Page



Review Statistics

Total Reviews: **170**

Positive Reviews: **101**

Negative Reviews: **69**

Did you find this dashboard helpful?

Thank you for your positive feedback! We're glad you found the dashboard helpful.

Figure 4.70 : Result Page.

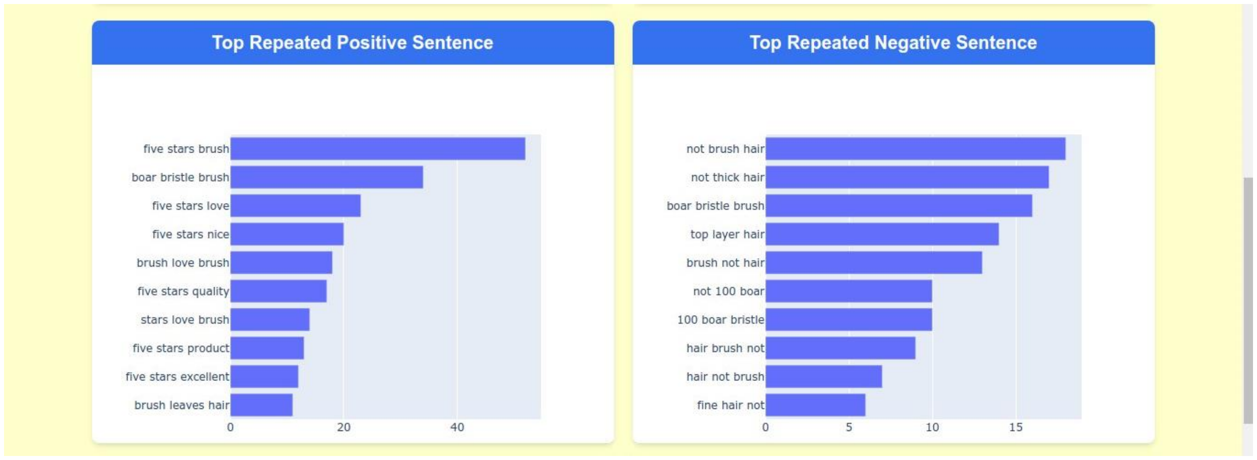
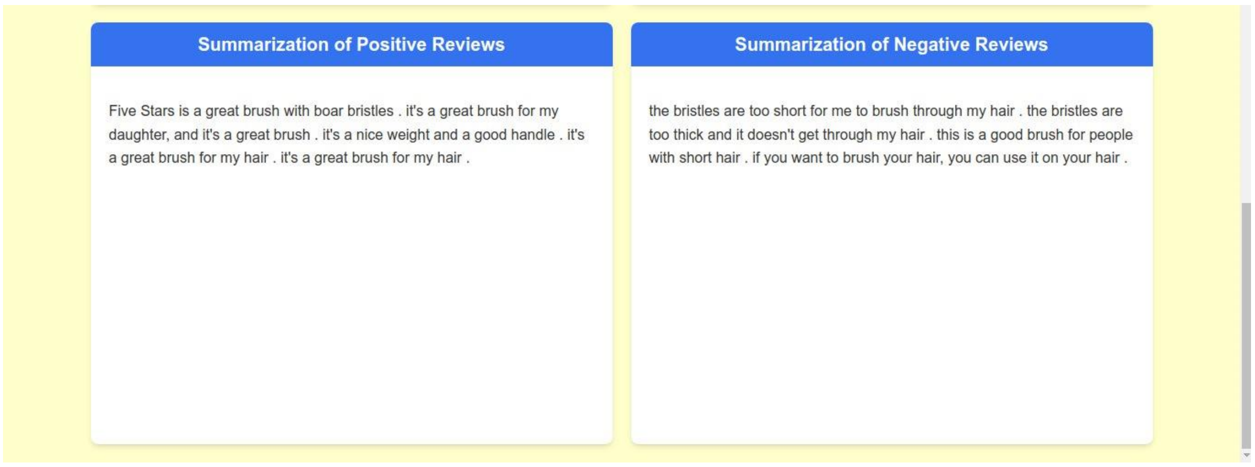


Figure 4.71 : Cont. Result Page.


Summarization of Positive Reviews
Summarization of Negative Reviews

Five Stars is a great brush with boar bristles . it's a great brush for my daughter, and it's a great brush . it's a nice weight and a good handle . it's a great brush for my hair . it's a great brush for my hair .

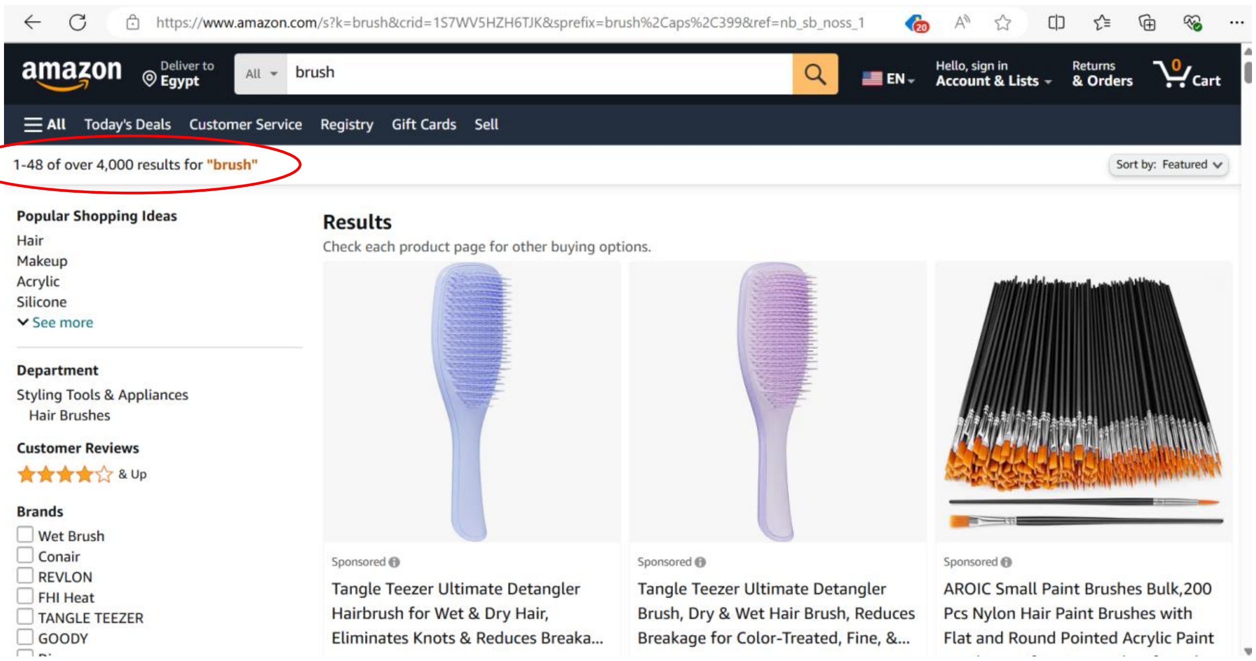
the bristles are too short for me to brush through my hair . the bristles are too thick and it doesn't get through my hair . this is a good brush for people with short hair . if you want to brush your hair, you can use it on your hair .

Figure 4.72 : Cont. Result Page.

Chapter 5:
IMPACT

5.1 Illustrative Example

Now, if you want to buy a product from Amazon.com, suppose it is a brush. When you search for it on Amazon website, you will have many results and you will be confused which one to choose. We will help you. Just take the link of any one of them and come to our website “Analytica”.



1-48 of over 4,000 results for "brush"

Results

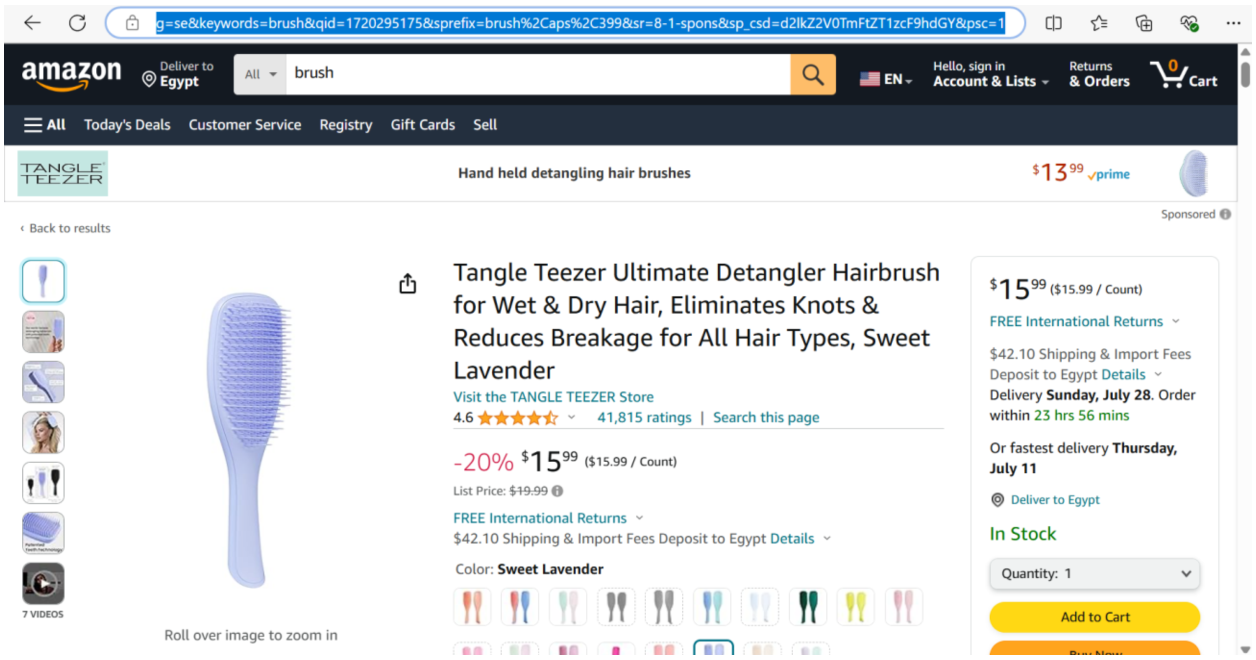
Check each product page for other buying options.

Tangle Teezer Ultimate Detangler Hairbrush for Wet & Dry Hair, Eliminates Knots & Reduces Breaka...

Tangle Teezer Ultimate Detangler Brush, Dry & Wet Hair Brush, Reduces Breakage for Color-Treated, Fine, &...

AROIC Small Paint Brushes Bulk,200 Pcs Nylon Hair Paint Brushes with Flat and Round Pointed Acrylic Paint

Figure 5.1 : Search on Amazon.com.



g=se&keywords=brush&qid=1720295175&sprefix=brush%2Caps%2C399&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1
amazon
Deliver to Egypt
All
brush
EN
Hello, sign in Account & Lists
Returns & Orders
0 Cart
All
Today's Deals
Customer Service
Registry
Gift Cards
Sell
TANGLE TEEZER
Hand held detangling hair brushes
$13.99 prime
Sponsored
Back to results
7 VIDEOS
Roll over image to zoom in
Add to Cart
Buy Now

Tangle Teezer Ultimate Detangler Hairbrush for Wet & Dry Hair, Eliminates Knots & Reduces Breakage for All Hair Types, Sweet Lavender

Visit the TANGLE TEEZER Store

4.6 41,815 ratings | Search this page

-20% $15.99 ($15.99 / Count)

List Price: $19.99

FREE International Returns

$42.10 Shipping & Import Fees Deposit to Egypt Details

Color: Sweet Lavender

$15.99 ($15.99 / Count)

FREE International Returns

$42.10 Shipping & Import Fees Deposit to Egypt Details

Delivery Sunday, July 28. Order within 23 hrs 56 mins

Or fastest delivery Thursday, July 11

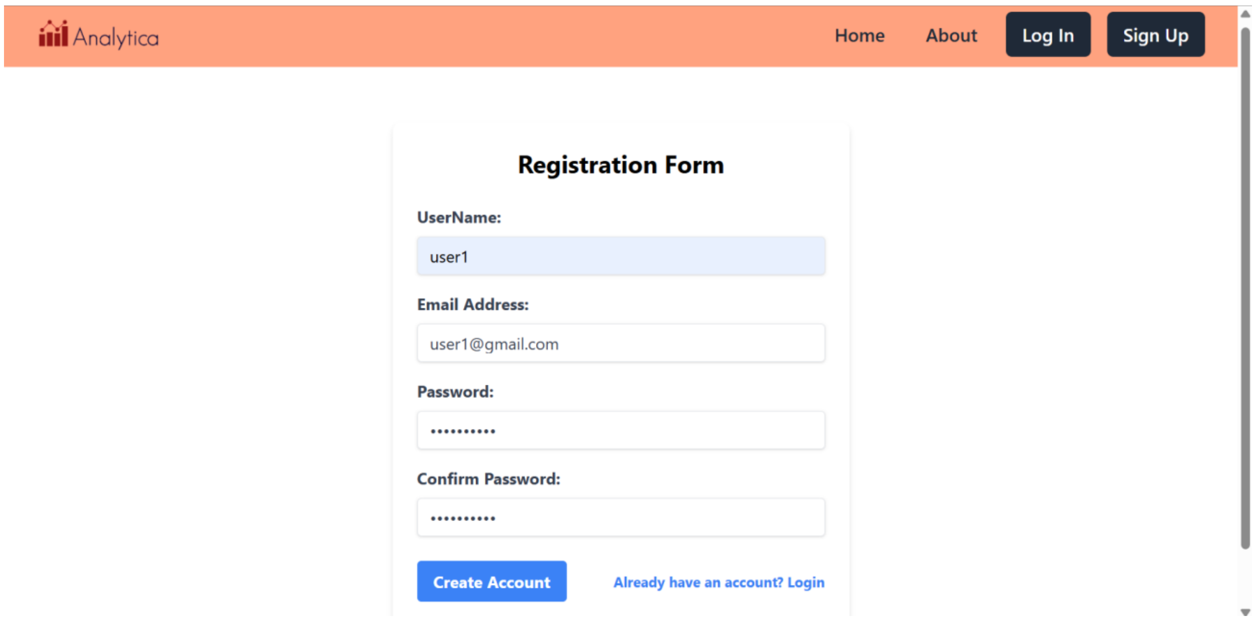
Deliver to Egypt

In Stock

Quantity: 1

Figure 5.2 : Take a product link.

First, if you have not an account on “Analytica” you should sign up to take a four attempts every day to try our website.



Registration Form

UserName: user1

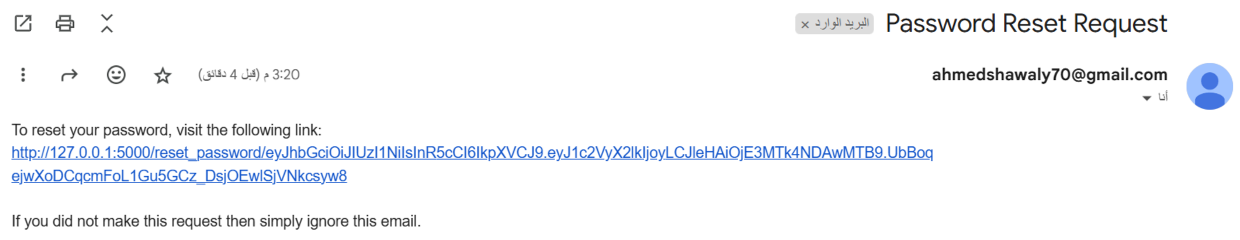
Email Address: user1@gmail.com

Password: ••••••••••

Confirm Password: ••••••••••

Figure 5.3 : Register.

Check the verification email which is send to you.



Password Reset Request

ahmedshawaly70@gmail.com
أنا

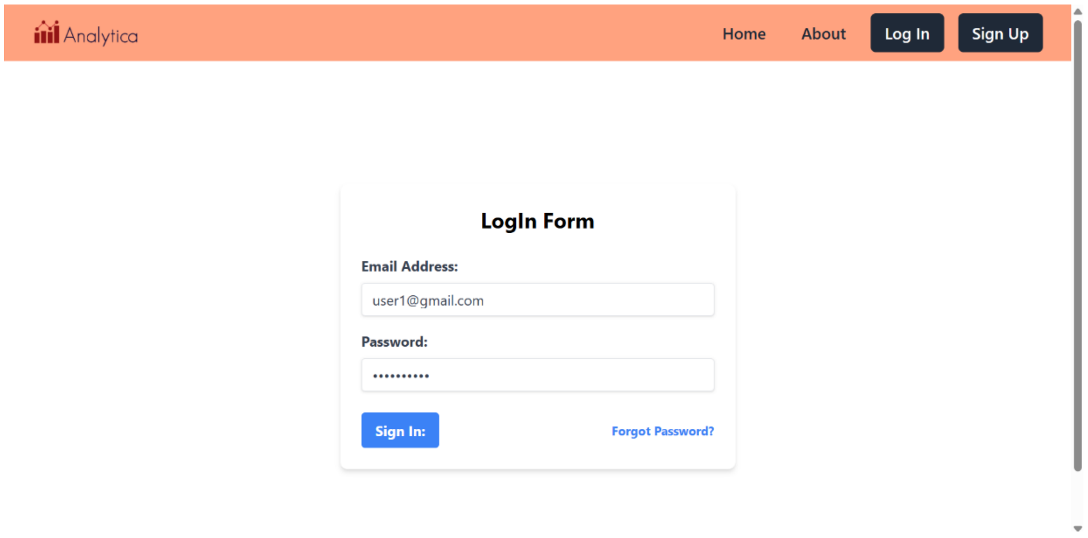
3:20 م (قبل 4 دقائق)

To reset your password, visit the following link:
http://127.0.0.1:5000/reset_password/eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoyLCJleHAiOjE3MTk4NDAwMTB9.UbBoq
ejwXoDCqcmFoL1Gu5GCz_DsjOEwlSjVNkcsyw8

If you did not make this request then simply ignore this email.

Figure 5.4 : Verification email.

If you already have an account just login.



LogIn Form

Email Address:

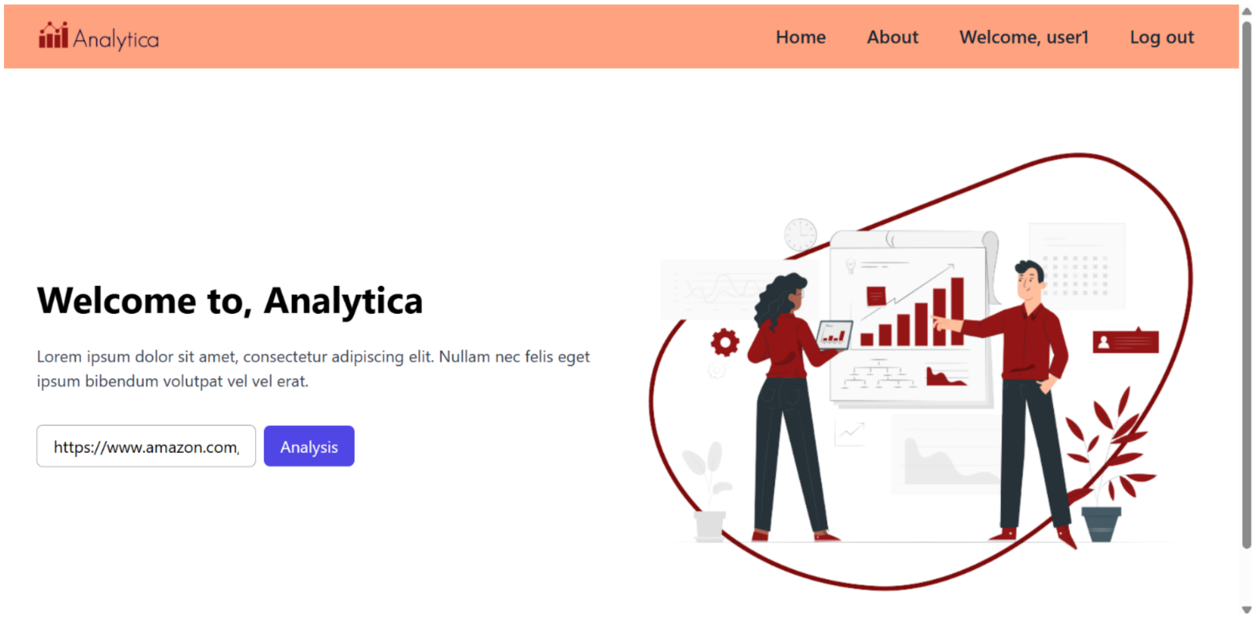
user1@gmail.com

Password:

••••••••••

Figure 5.5 : Login.

Now, you can enter the headphone link in the box designed for it and click analysis button.



Welcome to, Analytica

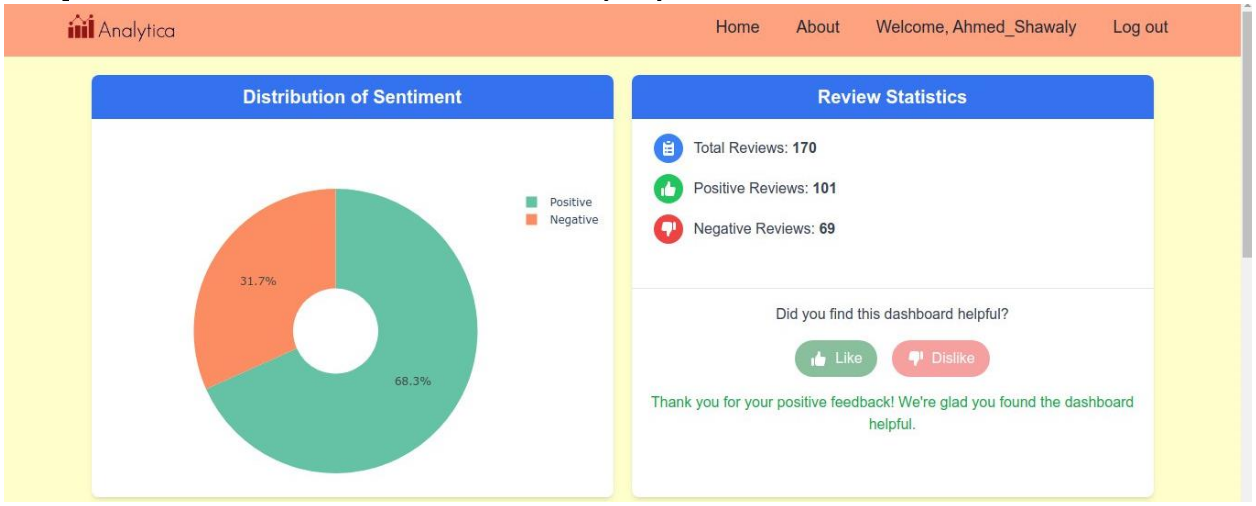
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam nec felis eget ipsum bibendum volutpat vel vel erat.

https://www.amazon.com,

Figure 5.6 : Put headphones link.

Wait a few moments and the results will be displayed.

You can be able to see sentiment analysis of people reviews who have already bought this headphones and tried it. You will know if the majority liked it or not.



Total Reviews: **170**

Positive Reviews: **101**

Negative Reviews: **69**

Did you find this dashboard helpful?

Thank you for your positive feedback! We're glad you found the dashboard helpful.

Figure 5.7 : Result Page.

Another features we offer is to know the top repeated sentences in positive and negative reviews separately.

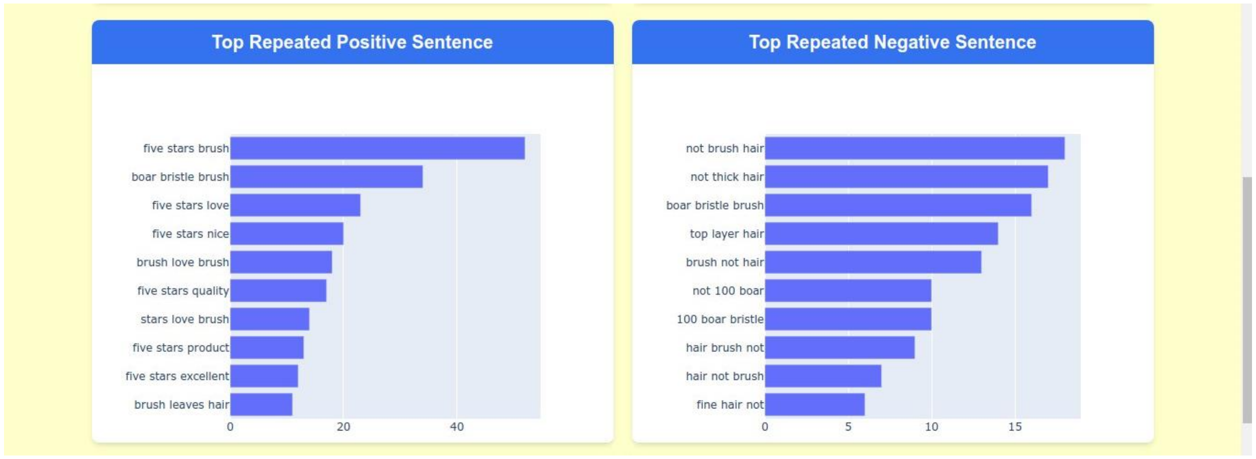
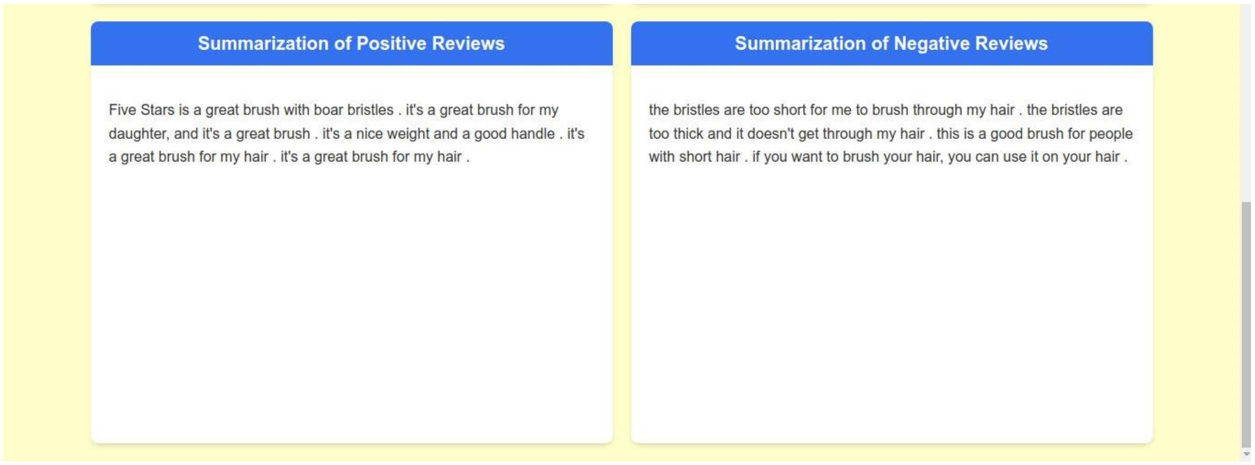


Figure 5.8 : Cont. Result Page.

The most important feature is a positive and negative reviews summarization, which save your time and tell you what is the advantages and disadvantages of this headphones without having to read a large number of reviews.


Summarization of Positive Reviews

Five Stars is a great brush with boar bristles . it's a great brush for my daughter, and it's a great brush . it's a nice weight and a good handle . it's a great brush for my hair . it's a great brush for my hair .

Summarization of Negative Reviews

the bristles are too short for me to brush through my hair . the bristles are too thick and it doesn't get through my hair . this is a good brush for people with short hair . if you want to brush your hair, you can use it on your hair .

Figure 5.9 : Result Page.

5.2 Target Audience

We target :

- Amazon.com.
- Retailers :
  They can pay to advertise their products in our website which is similar to what the user is searching for.
- Online buyers from Amazon :
  Any one wants to buy from Amazon and do not know if the product is worth it or not.

5.3 Reliability

5.3.1 Overview

The reliability of our software system was tested through many stages to ensure it meets the user needs and expectation

5.3.2 Testing methodology

We teste :

- That each component of the system individually to make sure that they work effectively (text preprocessing, tokenization and summarization).
- The interactions between all components to ensure they work together.
- The use case scenario to make sure that the overall functionality of the system is work and no any problem will face the user.
- Usability tests focused on the ease of use, easy to interface, and satisfaction with the generated summaries.

Chapter 6:

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Our software system is designed to provide user comfort and save his time. We offer some features (sentiment analysis, reviews classification, top repeated sentences, and reviews summarization) to meet user needs. RoBERTa (Robustly optimized BERT Approach) prove high efficiency as a transformer tool in sentiment analysis and reviews classification. T5-small is also effective transformer tool in reviews summarization. Our website provides a fast and reliable reviews classification and summarization.

6.2 Future Work

- Consider expanding the analysis to include product reviews from other e-commerce platforms such as Jumia, noon and souq.com.

- View recommendations depend on user activity history.

- Improve model accuracy.

- Develop a mobile app for the system.

Reference

[1] https://amazon-reviews-2023.github.io/index.html

[2] https://huggingface.co/datasets/fancyzhx/amazon_polarity

[3] https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/

[4] https://www.geeksforgeeks.org/overview-of-roberta-model/

[5] https://huggingface.co/docs/transformers/model_doc/distilbert

[6] https://www.geeksforgeeks.org/understanding-of-lstm-networks/

[7] alaBay94/Sentiment-analysis-amazon-Products-Reviews: NLP with NLTK for Sentiment analysis amazon Products Reviews (github.com)

[8] IBM/watson-second-opinion: Get a second opinion on Amazon products by analyzing product reviews with Watson Natural Language Understanding (github.com)

[9] https://github.com/Arjun-Mota/amazon-product-reviews-sentiment-analysis/blob/master/Amazon_reviews_sentiment_analysis.ipynb

[10] https://www.bing.com/search?pglt=43&q=T5-small&cvid=df83d03578b147ceabc64afde0ebb510&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIGCAEQABhAMgYIAhAAGEAyBggDEAAYQDIGCAQQABhAMgYIBRAAGEAyBggGEEUYPDIGCAcQRRg9MgYICBBFGDzSAQg2NzgxajBqMagCALACAA&FORM=ANNTA1&PC=U531

[11]https://www.researchgate.net/publication/379023578_A_novel_deep_learning_model_for_detection_of_inconsistency_in_e-commerce_websites

[12] https://matheo.uliege.be/bitstream/2268.2/2707/4/Memoire_MarieMartin_s112740.pdf

[13]https://www.researchgate.net/publication/378738833_Sentiment_analysis_applied_on_Amazon_reviews

[14]https://www.researchgate.net/publication/379456480_Analyzing_Amazon_Products_Sentiment_A_Comparative_Study_of_Machine_and_Deep_Learning_and_Transformer-Based_Techniques