



---

# Cloud Native AI & ML II : Assignment

---

OUMAIMA CHQAF

MASTER OF QUANTITATIVE AND FINANCIAL MODELLING  
MAJOR: DATA ENGINEERING

*Professor :*  
FAHD KALLOUBI

*Year :*  
2023/2024

# Contents

<b>1</b>	<b>Acknowledgement</b>	<b>3</b>
<b>2</b>	<b>Introduction</b>	<b>4</b>
<b>3</b>	<b>Using Mlflow</b>	<b>5</b>
3.1	Introduction . . . . .	5
3.2	Data preprocessing . . . . .	5
3.3	Model Training . . . . .	5
3.4	Mlflow implementation . . . . .	6
<b>4</b>	<b>Using FastAPI</b>	<b>7</b>
4.1	Serialization of Model and Preprocessing Transformations . . . . .	7
4.2	Development of REST API using FastAPI . . . . .	7
4.3	Consumption of APIs using Postman . . . . .	7
4.4	Conclusion and screens . . . . .	7
<b>5</b>	<b>Using Flask</b>	<b>10</b>
5.1	Setup and Implementation . . . . .	10
5.2	Flask Application Code . . . . .	10
5.3	User Interface and Interaction . . . . .	10
5.4	Deployment and Usage . . . . .	10
<b>6</b>	<b>Conclusion</b>	<b>13</b>

## List of Figures

1	Dataset IMDB Movie review : Overview of the feature it contains. . .	6
2	Shoot from our 'app.py' file. . . . .	8
3	Shoot from our 'app.py' file next part. . . . .	8
4	FastAPI in web browser. . . . .	8
5	Content of Postman. . . . .	9
6	Our file 'main.py'. . . . .	11
7	Home Page. . . . .	12
8	Entering the input. . . . .	12
9	Results. . . . .	13

# 1 Acknowledgement

I would like to take this opportunity to express my heartfelt appreciation to Professor Fahd Kalloubi, whose guidance, expertise, and unwavering support have been pivotal in shaping my journey through the "MLOps and Cloud Native AI/ML" module.

Professor Kalloubi's passion for the subject matter, combined with his exceptional teaching abilities, has not only deepened my understanding but has also ignited a profound interest in the convergence of Machine Learning Operations (MLOps) and Cloud Native AI/ML. His commitment to fostering a stimulating learning environment, marked by insightful lectures and practical demonstrations, has been instrumental in expanding my horizons within this rapidly evolving field.

## 2 Introduction

Sentiment analysis, a pivotal area in natural language processing (NLP), holds immense value in discerning sentiments, attitudes, and opinions conveyed within textual data. This report encapsulates a comprehensive exploration and deployment of sentiment analysis models tailored for a dataset comprising 50,000 movie reviews sourced from the IMDB dataset. The fundamental objective of this assignment revolves around the development and deployment of sentiment analysis models utilizing Flask and FastAPI, two renowned Python web frameworks. Our focus lies in predicting sentiment polarity, categorizing movie reviews into positive or negative sentiments, contributing to the broader landscape of sentiment analysis in the context of movie reviews.

The methodology adopted for this assignment traverses several integral stages. Commencing with data preprocessing and model training, we delve into the exploration and transformation of the IMDB movie review dataset, followed by the development and training of machine learning models using scikit-learn for sentiment analysis. Subsequently, our approach encompasses two distinct deployments using Flask and FastAPI. In the Flask deployment, we delineate the creation of a web application, offering a user-friendly interface where users can input movie reviews and receive sentiment predictions. Simultaneously, the FastAPI deployment focuses on constructing a robust RESTful API, empowering users to interact with the sentiment analysis model via API requests.

This report is structured to provide a comprehensive insight into the methodologies, challenges, and outcomes associated with the deployment of sentiment analysis models using Flask and FastAPI. The subsequent sections elucidate the data preprocessing and model training, delineate the intricacies of deploying sentiment analysis models through Flask and FastAPI frameworks, and conclude with a summary of key findings, challenges encountered, and potential avenues for further improvements and advancements in sentiment analysis methodologies.

## 3 Using Mlflow

### 3.1 Introduction

Sentiment analysis plays a pivotal role in understanding public opinion and sentiments towards various products, services, or in this case, movies. The objective of this assignment is to perform sentiment analysis on the IMDB dataset, comprised of 50,000 movie reviews. Utilizing machine learning models and Mlflow for model tracking and management, this report aims to evaluate different models for sentiment classification.

The IMDB dataset contains movie reviews labeled as positive or negative sentiments, making it an ideal dataset for sentiment analysis tasks. Leveraging Mlflow, an open-source platform for managing the end-to-end machine learning lifecycle, allows us to track models' performance, versions, and parameters efficiently.

### 3.2 Data preprocessing

The IMDB dataset underwent preprocessing steps essential for text data. This included:

- Text cleaning by removing HTML tags, special characters, and punctuation.
- Tokenization to split text into individual words.
- Removal of stop words and stemming to reduce feature dimensionality.
- Splitting the dataset into a 70-30 ratio for training and testing respectively.

### 3.3 Model Training

The Logistic Regression model was chosen as one of the primary models for sentiment analysis due to its simplicity and interpretability. Below outlines the steps taken for its training:

- **Model Selection:** Logistic Regression, a linear classification algorithm, was chosen for its efficiency in handling text-based classification tasks.
- **Feature Engineering - Bag-of-Words (BoW):**
  - The text data underwent preprocessing steps including text cleaning, tokenization, and creating a Bag-of-Words representation.
  - BoW representation was generated using the CountVectorizer from scikit-learn.
- **Model Training:** The BoW representation derived from the preprocessed text data was used to train the Logistic Regression model

Data:-

- **Poster\_Link** - Link of the poster that imdb using
- **Series\_Title** - Name of the movie
- **Released\_Year** - Year at which that movie released
- **Certificate** - Certificate earned by that movie
- **Runtime** - Total runtime of the movie
- **Genre** - Genre of the movie
- **IMDB\_Rating** - Rating of the movie at IMDB site
- **Overview** - mini story/ summary
- **Meta\_score** - Score earned by the movie
- **Director** - Name of the Director
- **Star1,Star2,Star3,Star4** - Name of the Stars
- **No\_of\_votes** - Total number of votes
- **Gross** - Money earned by that movie

Figure 1: Dataset IMDB Movie review : Overview of the feature it contains.

- **Model Evaluation:** The trained Logistic Regression model was evaluated on the test dataset to assess its performance using various evaluation metrics such as accuracy, precision, recall, and F1-score.

### 3.4 Mlflow implementation

Mlflow, an open-source platform, was instrumental in managing the end-to-end machine learning lifecycle, enabling efficient tracking of model development, experimentation, and performance. The integration of Mlflow facilitated comprehensive monitoring of models, versions, and associated parameters throughout the training process.

## 4 Using FastAPI

### 4.1 Serialization of Model and Preprocessing Transformations

The trained Logistic Regression model for sentiment analysis was serialized into the ONNX format, ensuring its portability and compatibility across different platforms. Additionally, the preprocessing transformations essential for text data preprocessing were serialized and saved in a pickle format. These serialized files served as crucial components for building the REST API and performing inference tasks.

### 4.2 Development of REST API using FastAPI

Integration of Serialized Model and Preprocessing Transformations FastAPI, a modern web framework for building APIs with Python, was employed to create a RESTful service for sentiment analysis. The FastAPI application was configured to load the serialized Logistic Regression model stored in ONNX format and the preprocessing transformations saved in pickle format.

Implementation of Prediction Endpoint An HTTP GET endpoint named `/predict` was defined within the FastAPI application to receive incoming requests containing movie reviews. Upon receiving a review as input, the endpoint invoked the loaded model and preprocessing transformations to perform sentiment analysis. The predicted sentiment (positive/negative) for the provided review was returned as the response.

### 4.3 Consumption of APIs using Postman

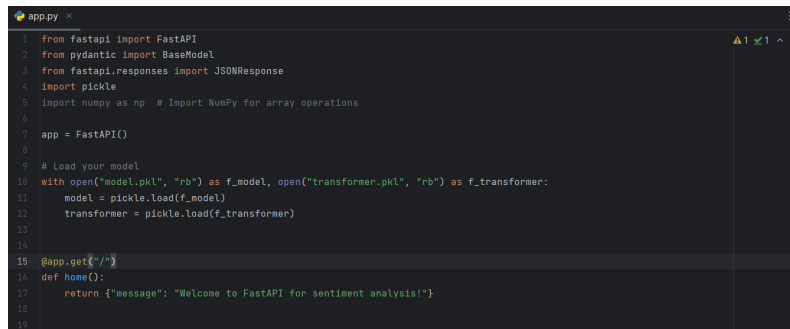
Usage of Postman for API Testing Postman, a popular API development and testing tool, was employed to consume and test the functionality of the created REST API. Requests were made to the API endpoints, particularly the `/predict` endpoint, by providing movie reviews as query parameters. Postman facilitated the visualization of API responses, allowing validation of the sentiment predictions generated by the deployed model.

### 4.4 Conclusion and screens

Here is the code we wrote for FastAPI. Now let's use Postman to try and see the results of our app. Here is what we see in the web browser before using Postman:

Now let's take a look on what we wrote in Postman. As you can see from the figure above, we gave our API a review and returned the prediction of the sentiment in that case it was 'positive'.



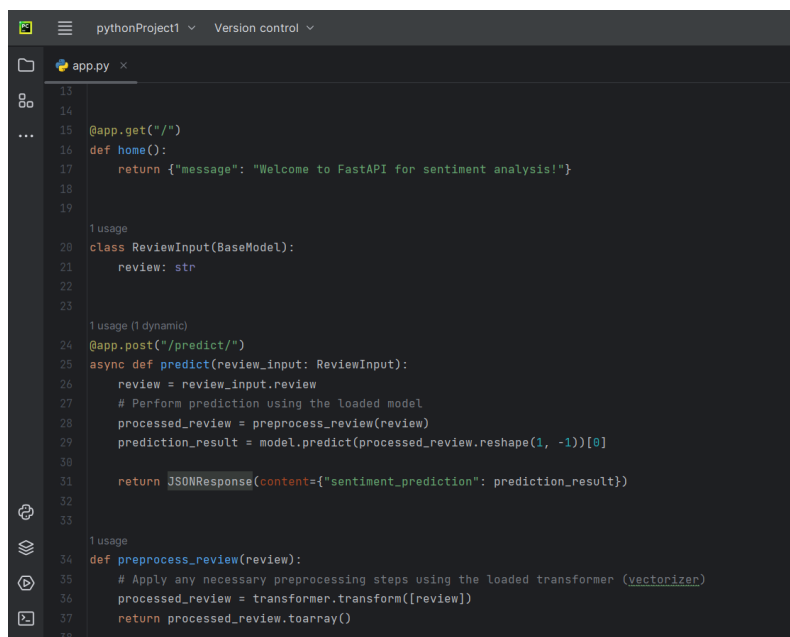


```

1 from fastapi import FastAPI
2 from pydantic import BaseModel
3 from fastapi.responses import JSONResponse
4 import pickle
5 import numpy as np # Import NumPy for array operations
6
7 app = FastAPI()
8
9 # Load your model
10 with open("model.pkl", "rb") as f_model, open("transformer.pkl", "rb") as f_transformer:
11     model = pickle.load(f_model)
12     transformer = pickle.load(f_transformer)
13
14
15 @app.get("/")
16 def home():
17     return {"message": "Welcome to FastAPI for sentiment analysis!"}
18
19

```

Figure 2: Shoot from our 'app.py' file.

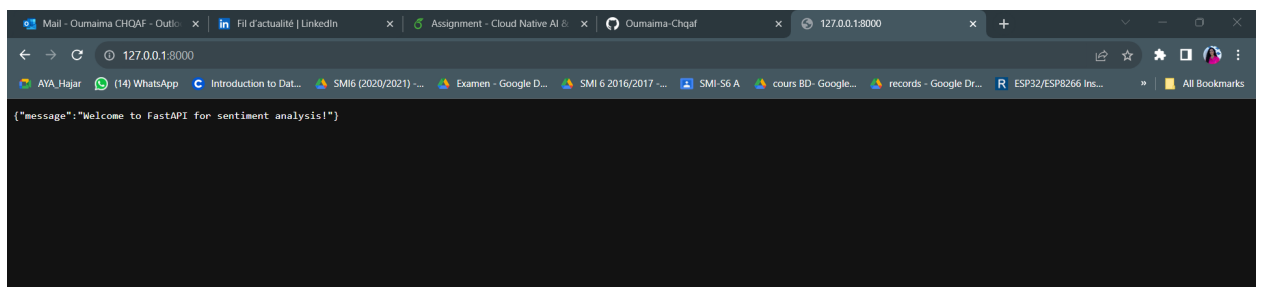


```

13
14
15 @app.get("/")
16 def home():
17     return {"message": "Welcome to FastAPI for sentiment analysis!"}
18
19
20 class ReviewInput(BaseModel):
21     review: str
22
23
24 @app.post("/predict/")
25 async def predict(review_input: ReviewInput):
26     review = review_input.review
27     # Perform prediction using the loaded model
28     processed_review = preprocess_review(review)
29     prediction_result = model.predict(processed_review.reshape(1, -1))[0]
30
31     return JSONResponse(content={"sentiment_prediction": prediction_result})
32
33
34 def preprocess_review(review):
35     # Apply any necessary preprocessing steps using the loaded transformer (vectorizer)
36     processed_review = transformer.transform([review])
37     return processed_review.toarray()
38

```

Figure 3: Shoot from our 'app.py' file next part.



```

{"message": "Welcome to FastAPI for sentiment analysis!"}

```

Figure 4: FastAPI in web browser.

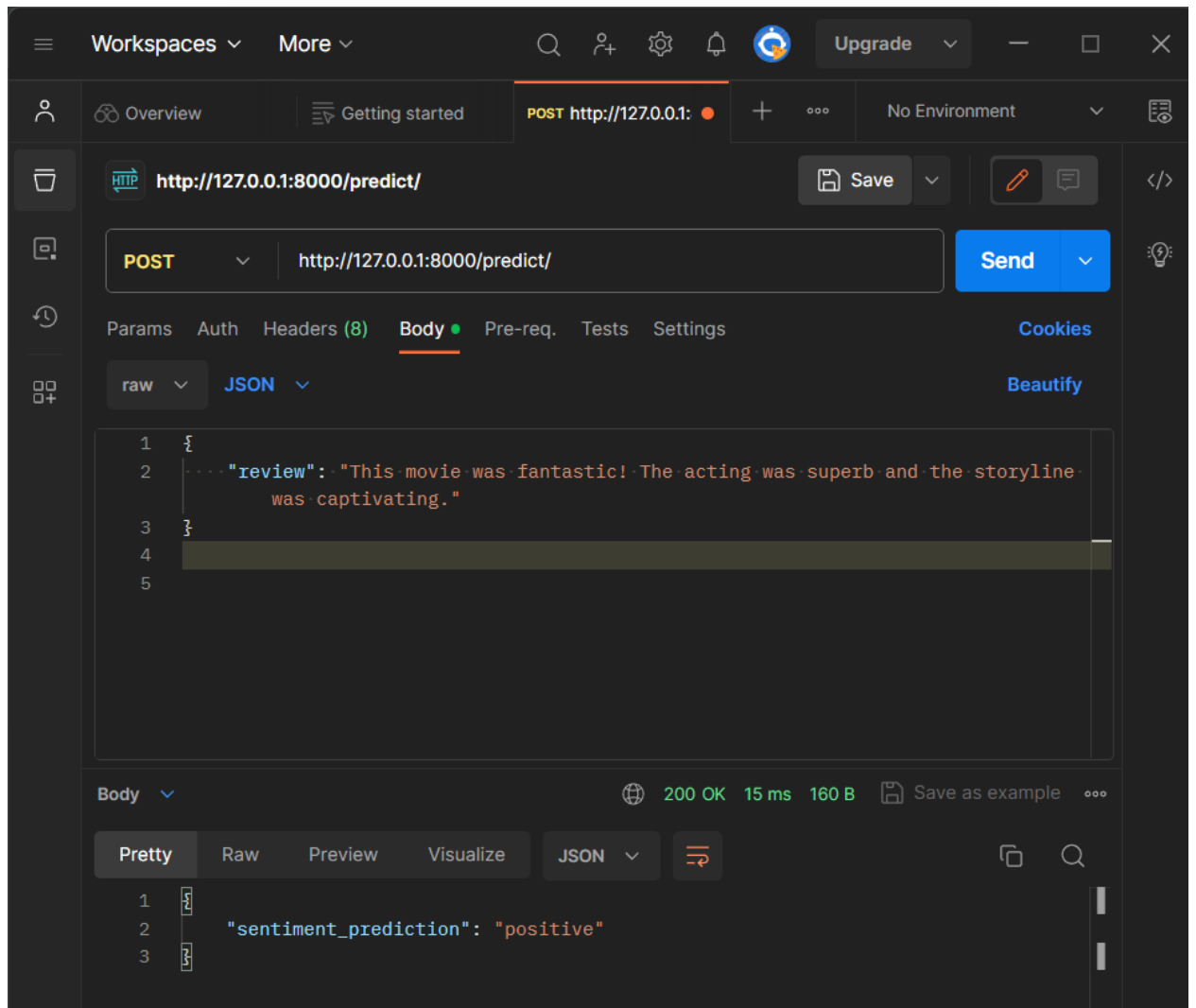


Figure 5: Content of Postman.

## 5 Using Flask

In this phase of the project, we deployed a sentiment analysis model using Flask, a micro web framework for Python, to create a basic web application capable of analyzing the sentiment of movie reviews.

### 5.1 Setup and Implementation

- **Installation and Environment Setup:** Initially, we installed the required Python libraries, including Flask, scikit-learn, and other necessary dependencies for the sentiment analysis model.
- **Development of Sentiment Analysis Model:** We trained a sentiment analysis model using scikit-learn, which was serialized and saved as a pickle file (model.pkl). Additionally, we stored the preprocessing transformer (e.g., CountVectorizer or TfidfVectorizer) as transformer.pkl.
- **Creation of Flask Application:** We developed a Flask application (app.py) consisting of several routes to handle various functionalities of the sentiment analysis service.

### 5.2 Flask Application Code

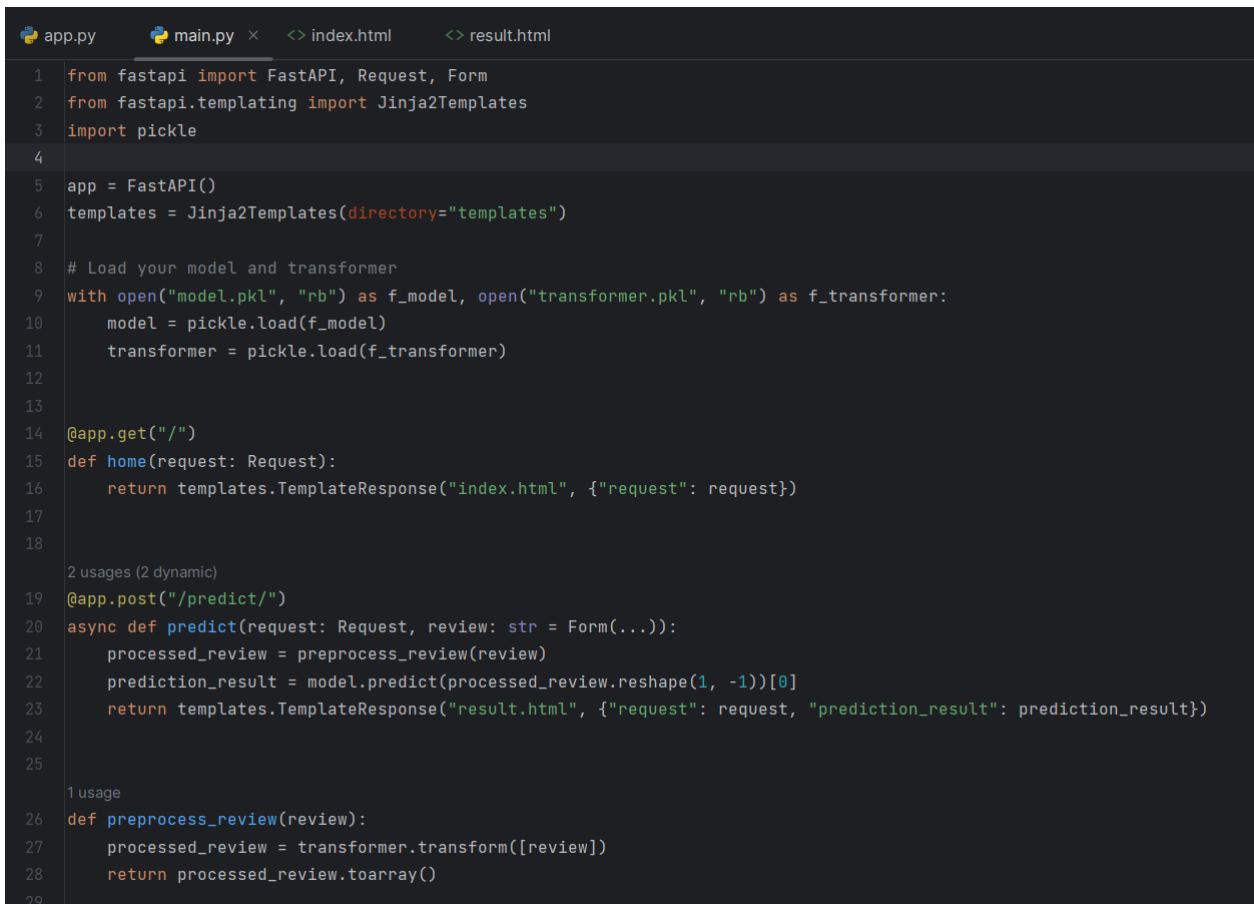
The main.py file consisted of the following components:

### 5.3 User Interface and Interaction

- **User-Friendly Interface:** The Flask application provided a basic yet user-friendly interface. Upon accessing the root route ("/"), users were presented with an HTML form to input their movie reviews.
- **Sentiment Analysis Functionality:** After entering a review and submitting the form, the input was sent to the backend for sentiment analysis. The Flask server performed sentiment prediction using the loaded model and responded with the predicted sentiment.

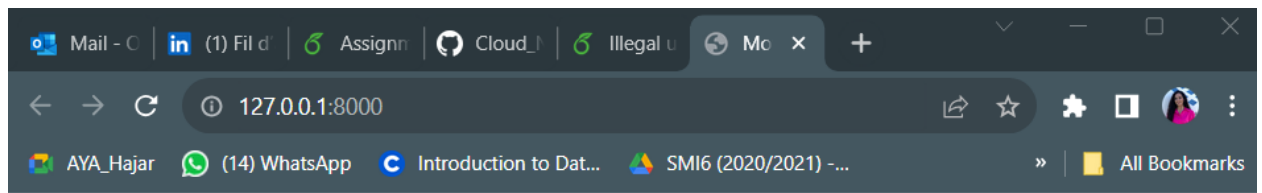
### 5.4 Deployment and Usage

- **Local Deployment:** The Flask application was locally deployed by running the app.py file. Users accessed the sentiment analysis service by navigating to http://localhost:5000 in their web browser.
- **Usage:** Users interacted with the interface by entering movie reviews, and the application provided sentiment predictions based on the input.



```
1 from fastapi import FastAPI, Request, Form
2 from fastapi.templating import Jinja2Templates
3 import pickle
4
5 app = FastAPI()
6 templates = Jinja2Templates(directory="templates")
7
8 # Load your model and transformer
9 with open("model.pkl", "rb") as f_model, open("transformer.pkl", "rb") as f_transformer:
10     model = pickle.load(f_model)
11     transformer = pickle.load(f_transformer)
12
13
14 @app.get("/")
15 def home(request: Request):
16     return templates.TemplateResponse("index.html", {"request": request})
17
18
19 2 usages (2 dynamic)
20 @app.post("/predict/")
21 async def predict(request: Request, review: str = Form(...)):
22     processed_review = preprocess_review(review)
23     prediction_result = model.predict(processed_review.reshape(1, -1))[0]
24     return templates.TemplateResponse("result.html", {"request": request, "prediction_result": prediction_result})
25
26 1 usage
27 def preprocess_review(review):
28     processed_review = transformer.transform([review])
29     return processed_review.toarray()
```

Figure 6: Our file 'main.py'.

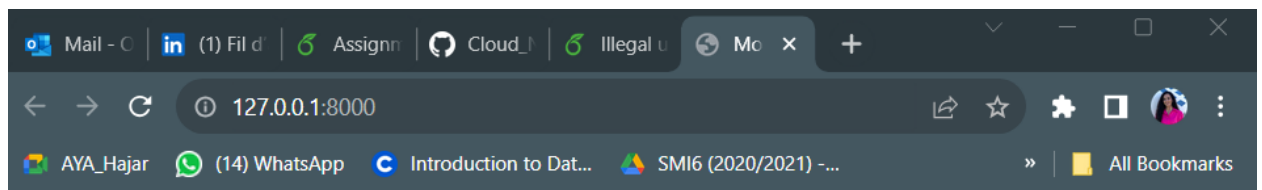


# Welcome to Movie Review Sentiment Analysis!

Enter your movie review:

Submit

Figure 7: Home Page.

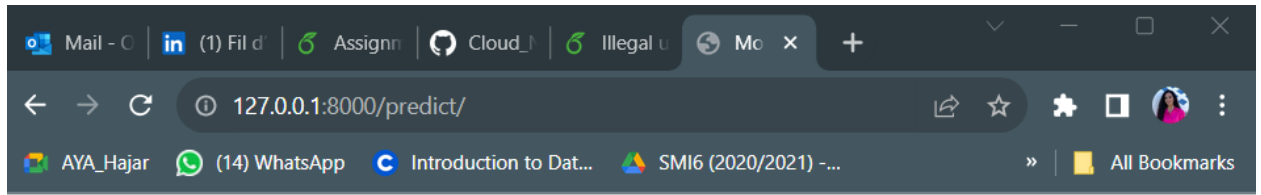


# Welcome to Movie Review Sentiment Analysis!

Enter your movie review:

Submit

Figure 8: Entering the input.



## Sentiment Prediction

The sentiment prediction for the provided movie review is: **positive**

[Go back to enter another review](#)

Figure 9: Results.

## 6 Conclusion

In conclusion, the deployment and exploration of sentiment analysis models using Flask and FastAPI have illuminated the diverse capabilities and nuances within the realm of natural language processing, specifically in deciphering sentiments embedded within movie reviews. Through meticulous data preprocessing and model training, we successfully developed machine learning models adept at discerning sentiments from textual data, laying the foundation for accurate predictions of positive or negative sentiments associated with movie reviews.

The utilization of Flask and FastAPI as deployment frameworks offered unique insights into the varied methodologies of creating sentiment analysis services. The Flask implementation provided an intuitive and user-friendly web interface, enabling seamless interactions for users to input movie reviews and obtain sentiment predictions. On the other hand, FastAPI demonstrated its prowess in building a robust and efficient RESTful API, facilitating access to sentiment analysis functionalities through API requests, thus catering to diverse user needs and preferences.

Despite the accomplishments, this assignment also brought forth notable challenges. Addressing issues related to model scalability, optimizing API performance, and refining user interfaces emerged as key considerations for future enhancements. Moreover, the continuous evolution of NLP techniques and advancements in model architectures present exciting opportunities for enhancing sentiment analysis capabilities, warranting further research and development in this domain.

In essence, this assignment not only showcased the successful deployment of sentiment analysis models but also underscored the ongoing potential and need for continued advancements in NLP methodologies. The strides made herein lay the groundwork for further innovations, setting the stage for future endeavors aimed at

refining and expanding the horizons of sentiment analysis in diverse applications.